

AFIT/DS/ENS/97-01

NESTED FORK-JOIN QUEUING
NETWORKS AND THEIR APPLICATION
TO MOBILITY AIRFIELD OPERATIONS
ANALYSIS

DISSERTATION

Craig Joslyn Willits
Major, USAF

AFIT/DS/ENS/97-01

19970403 061

DEPT OF DEFENSE

Approved for public release; distribution unlimited

The views expressed in this dissertation are those of the author and do not reflect the official policy of the Department of Defense or the U. S. Government.

AFIT/DS/ENS/97-01

NESTED FORK-JOIN QUEUING NETWORKS
AND THEIR APPLICATION TO MOBILITY AIRFIELD
OPERATIONS ANALYSIS

DISSERTATION

Presented to the Faculty of the Graduate School of Engineering
of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the
Requirements for the Degree of
Doctor of Philosophy

Craig Joslyn Willits, B.A., M.S.

Major, USAF

March 1997

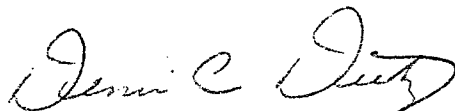
Approved for public release; distribution unlimited

NESTED FORK-JOIN QUEUING NETWORKS
AND THEIR APPLICATION TO MOBILITY AIRFIELD
OPERATIONS ANALYSIS

Craig Joslyn Willits, B.A., M.S.
Major, USAF

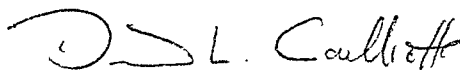
Approved:

Date



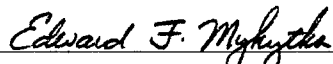
7 Mar 97

Dennis C. Dietz, Ph.D. (Chairman)



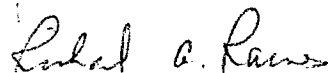
7 Mar 97

David L. Coulliette, Ph.D.



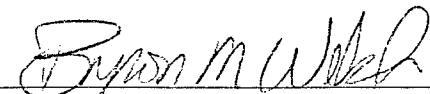
7 Mar 97

Edward F. Mykytka, Ph.D.



7 Mar 97

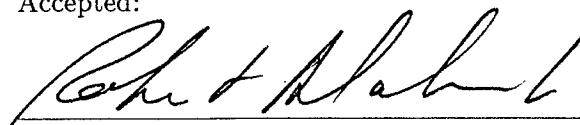
Richard A. Raines, Ph.D.



7 Mar 97

Byron M. Welsh, Ph.D. (Dean's Representative)

Accepted:



ROBERT A. CALICO, Jr., Ph.D.
Dean, Graduate School of Engineering

Acknowledgements

I do not know what I may appear to the world; but to myself I seem to have been only like a boy playing on the sea-shore, and diverting myself in now and then finding a smoother pebble or a prettier shell than ordinary, whilst the great ocean of truth lay all undiscovered before me.

Sir Isaac Newton

In the process of conducting this research, I have indeed felt the joy of discovery Newton so aptly described. Unlike Newton, however, I was not alone on the beach of discovery.

Of course, I would not have succeeded in my efforts without my committee's contributions. Dr Dennis Dietz, my research advisor, has been my mentor and counselor since I began work on my master's thesis nearly four years ago; his importance to the success of this project cannot be overstated. I am especially grateful to Dr Dave Coulliette for lending his expertise in numerical analysis; his support was invaluable during the preparation of Chapter III. My thanks go to Dr Ed Mykytka for asking the hard questions which led to a sharper written presentation. Dr Rick Raines brought a fresh perspective that I appreciated. Dr Byron Welsh, the Dean's representative, and Dr Pete Hovey, a former committee member, deserve special mention for their helpful suggestions.

Besides my committee, I was fortunate to have outstanding support from Air Mobility Command (AMC), which sponsored my research. Dr Jean Steppe had the original vision for this project; her advice and suggestions helped me keep my approach balanced between theory and practice. Dr John Borsi increased my understanding of mobility modeling issues. Kim Schubert, Travis Cusick and Alan Whisman provided valuable insights into the details of AMC's mobility airfield simulation models.

I am indebted to several friends and colleagues for their insight, empathy and emotional support. My current and former office mates, John Van Hove and Dr Dan Zalewski, deserve credit for enduring the times I waxed eloquent about queuing theory. Fellow student Pete Vanden Bosch and I had many fruitful discussions about the nature of phase-type probability distributions. My colleagues Robert Brigantic and Steve Forsythe empathized with many of my frustrations, especially with balky computer code. Dr Al Moore, another of my mentors, encouraged me with his interest in my progress. The prayers and support of my faithful friends Milt Barney, Darren Gibbs, Dr Mark Oxley, and Phil Shaw were much appreciated, especially in the days leading up to the completion of this manuscript.

As always, my wife Sara was my chief source of encouragement and the leader of my support team. I above all appreciate her willingness to listen as I dealt with the frustrations of original research. I am also thankful for my sons, Steven, David, and Joseph, who love me simply because I'm their Dad, and are not impressed with dissertations and degrees.

Finally, I give all praise to God, and dedicate this dissertation to His glory. "A man's heart deviseth his way: but the Lord directeth his steps" (Proverbs 16:9).

Craig Joslyn Willits

Table of Contents

	Page
Acknowledgements	iii
Table of Contents	v
List of Figures	x
List of Tables	xiii
Abstract	xvi
 I. Introduction	 1
Motivation	1
Problem Statement	2
The Analytical Airfield Model	3
Overview.	3
The Network Structure.	3
Customer Classes.	7
Network Capacity.	8
The Arrival Processes.	8
The Service Stations.	8
Research Goals	9
Sequence of Presentation	9
 II. Analyzing Queuing Networks	 10
Overview	10
Basic Concepts	10

	Page
Product-Form Approximation of Closed Queuing Networks . .	12
Background.	12
Aggregation.	13
Marie's Method.	14
Analyzing Multiserver Stations Using Marie's Method	17
III. Efficient Isolated Analysis of General Multiserver Stations	
During Decomposition of Closed Queuing Networks	20
Introduction and Motivation	20
The Transition Rate Matrix for the $\lambda(n)/C_k/r/N$ Queue	21
Constructing the Matrix.	21
The Structure of the Matrix.	24
Solution Methods: Narrowing the Choice	26
Computational Experience	27
The Candidate Iterative Methods.	27
The Representative Queuing Systems.	27
Computing Environment.	29
Numerical Analytical Considerations.	29
Results	30
Using CGS for Isolated Analysis in Marie's Method	39
Conclusions	44
IV. Fork-Join Queuing Networks	46
Introduction	46
Network Types	46
The Fork-Join Queue.	46
Networks With Fork-Join Primitives.	48
Review of the Literature	49

	Page
Motivation.	49
Simple Fork-Join Queues.	49
Open Networks With Fork and Join Primitives.	56
Closed Networks Containing Fork-Join Subnetworks. . .	58
Discussion of the Literature.	61
Product-Form Approximation of Closed Fork-Join Queuing Net- works.	62
Introduction.	62
Analysis Using Aggregation.	62
Analysis Using Marie's Method.	64
Necessary Theoretical Extensions	65
V. A Product-Form Approximation Technique for Closed Nested Fork-Join Queuing Networks With Probabilistic Load Patterns	67
Introduction	67
The "Short-Circuit" Approximation	67
Description.	67
Analyzing the Synchronization Station.	68
Extension to Nested Fork-Join Queuing Networks.	74
Computational Experience	76
Case Study 1: FJQNs With Probabilistic Forking.	76
Case Study 2: FJQNs With Nested FJSNs.	89
Conclusion	97
Recommendations	99
VI. Analytical Airfield Model Demonstration	101
Introduction	101
Implementing the Analytical Airfield Model	101
Numerical Study	104

	Page
Overview.	104
Results.	104
Error Analysis.	109
Speed of Execution.	109
Conclusion and Recommendations	109
VII. Conclusion	111
Introduction	111
Statement of Research Contributions	111
Analysis of Multiserver k -Coxian Queues.	111
Decomposition of Nested Fork-Join Queuing Networks.	111
Isolated Analysis of Mobility Airfield Flow.	111
Summary of Recommended Research Topics	112
Appendix A. Calculating the Stationary Probabilities of a Continuous-Time Markov Chain	114
Introduction	114
The Stationary Probability Distribution	114
Definition.	114
Problem Stability.	115
Algebraic Solution of the Stationary Probability Problem	116
Direct Solution Methods for Linear Systems.	117
Iterative Solution Methods.	118
Solving the Stationary Probability Problem Using Decomposition	125
Solving the Stationary Probability Problem Using Recursion	127
Appendix B. The $\lambda(n)/C_k/r/N$ Queue: Numerical Results for Generalized k -Erlang Systems	129
Appendix C. Nested FJQNs: Numerical Results	135

	Page
Appendix D. The Analytical Airfield Model: Numerical Results	154
Bibliography	160
Vita	171

List of Figures

Figure	Page
1. Task Precedence Graph.	4
2. Graph of the Equivalent Queuing Network.	6
3. The Product-Form Approximation Technique.	13
4. Surface Contour Plot of m versus k and r , $N = 30$	23
5. Block Structure of Q	25
6. k -Erlang Systems: Solution Time vs. Order, No Preconditioning. . .	33
7. k -Erlang Systems: Solution Time vs. Order, ILU(0) Preconditioner. .	34
8. k -Erlang Systems: Factorization Time vs. Order, ILU(0) Preconditioner.	35
9. k -Erlang Systems: Iteration Time vs. Order, ILU(0) Preconditioner. .	36
10. Memory Usage vs. Order.	37
11. General Network Topology.	41
12. Equivalent Topology, Networks Ia and Ib.	43
13. Equivalent Topology, Networks IIa and IIb.	43
14. Sample Fork-Join Queue Topology.	47
15. Sample PFJQN Topology.	48
16. Sample AFJQN Topology.	49
17. Representative Fork-Join Queuing Network.	63
18. Isolated Fork-Join Subnetwork, Aggregation Method.	63
19. Transformed Subnetwork, Aggregation Method.	64
20. Isolated Fork-Join Subnetwork, Marie's Method.	65
21. Transformed Subnetwork, Marie's Method.	66
22. Using SC Approximation With Marie's Method.	69
23. Hierarchical Decomposition of a Nested FJQN.	75
24. Basic Network Topology, Case Study 1 Representative Systems. . . .	77

Figure	Page
25. Case Study 1: Expected Throughput at Station 1, Systems 1-9. . . .	81
26. Case Study 1: Expected Throughput at Station 1, Systems 10-18. .	81
27. Case Study 1: Expected Queue Length at Station 1, Systems 1-9. . .	82
28. Case Study 1: Expected Queue Length at Station 1, Systems 10-18.	82
29. Case Study 1: Expected Queue Length at Station 2, Systems 1-9. . .	83
30. Case Study 1: Expected Queue Length at Station 2, Systems 10-18.	83
31. Case Study 1: Expected Queue Length at Station 3, Systems 1-9. . .	84
32. Case Study 1: Expected Queue Length at Station 3, Systems 10-18.	84
33. Case Study 1: Expected Queue Length at Station 4, Systems 1-9. . .	85
34. Case Study 1: Expected Queue Length at Station 4, Systems 10-18.	85
35. Case Study 1: Expected Queue Length at Station 5, Systems 1-9. . .	86
36. Case Study 1: Expected Queue Length at Station 5, Systems 10-18.	86
37. Case Study 1: Expected Queue Length at Station 6, Systems 1-9. . .	87
38. Case Study 1: Expected Queue Length at Station 6, Systems 10-18.	87
39. Case Study 1: Expected Queue Length at Station 7, Systems 1-9. . .	88
40. Case Study 1: Expected Queue Length at Station 7, Systems 10-18.	88
41. Basic Network Topology, Case Study 2 Representative Systems. . . .	90
42. Case Study 2: Expected Throughput at Station 1.	91
43. Case Study 2: Expected Queue Length at Station 1.	92
44. Case Study 2: Expected Queue Length at Station 2.	92
45. Case Study 2: Expected Queue Length at Station 3.	93
46. Case Study 2: Expected Queue Length at Station 4.	93
47. Case Study 2: Expected Queue Length at Station 5.	94
48. Case Study 2: Expected Queue Length at Station 6.	94
49. Case Study 2: Expected Queue Length at Station 7.	95
50. Case Study 2: Expected Queue Length at Station 8.	95
51. Case Study 2: Expected Queue Length at Station 9.	96

Figure	Page
52. Case Study 2: Expected Queue Length at Station 10.	96
53. Revised AAM Topology (Stations 3, 4, and 10 Removed).	102
54. Effect of Mean Interarrival Time on Airfield Throughput.	105
55. Effect of Mean Interarrival Time on Airfield Response Time.	106
56. Effect of Mean Interarrival Time on Aircraft on Station.	106
57. Effect of Constrained Resources on Airfield Throughput.	107
58. Effect of Constrained Resources on Airfield Response Time.	108
59. Effect of Constrained Resources on Aircraft on Station.	108
60. Generalized k -Erlang Systems: Solution Time vs. Order, No Preconditioning.	131
61. Generalized k -Erlang Systems: Solution Time vs. Order, ILU(0) Preconditioner.	132
62. Generalized k -Erlang Systems: Factorization Time vs. Order, ILU(0) Preconditioner.	133
63. Generalized k -Erlang Systems: Iteration Time vs. Order, ILU(0) Preconditioner.	134

List of Tables

Table	Page
1. Network Station Descriptions.	5
2. Ordering of States When $k = 4$, $r = 3$, and $3 < n < N$	22
3. Representative System Configurations.	28
4. Parameters of the Generalized k -Erlang Distributions.	28
5. Elapsed Time to Solution (in seconds), No Preconditioner.	31
6. Elapsed Time to Solution (in seconds), ILU(0) Preconditioner.	32
7. Convergence Behavior, Arrival Process II, No Preconditioner.	40
8. Parameters of Stations 9 and 10.	42
9. Results of Marie's Method Decomposition.	44
10. State Transitions, Aggregation.	71
11. Column Entries For Row s of Q , Aggregation.	73
12. State Transitions, Marie's Method.	73
13. Column Entries For Row s of Q , Marie's Method.	74
14. Case Study 1: Representative System Configurations.	78
15. Case Study 2: Representative System Configurations.	89
16. Case Study 2: Convergence Summary.	98
17. Analytical Airfield Model Station Descriptions.	103
18. Elapsed Time to Solution (in seconds), No Preconditioner.	129
19. Elapsed Time to Solution (in seconds), ILU(0) Preconditioner.	130
20. Case Study 1: Throughput at Station 1.	135
21. Case Study 1: Queue Lengths at Station 1.	136
22. Case Study 1: Queue Lengths at Station 2.	137
23. Case Study 1: Queue Lengths at Station 3.	138
24. Case Study 1: Queue Lengths at Station 4.	139

Table	Page
25. Case Study 1: Queue Lengths at Station 5.	140
26. Case Study 1: Queue Lengths at Station 6.	141
27. Case Study 1: Queue Lengths at Station 7.	142
28. Case Study 2: Throughput at Station 1.	143
29. Case Study 2: Queue Lengths at Station 1.	144
30. Case Study 2: Queue Lengths at Station 2.	145
31. Case Study 2: Queue Lengths at Station 3.	146
32. Case Study 2: Queue Lengths at Station 4.	147
33. Case Study 2: Queue Lengths at Station 5.	148
34. Case Study 2: Queue Lengths at Station 6.	149
35. Case Study 2: Queue Lengths at Station 7.	150
36. Case Study 2: Queue Lengths at Station 8.	151
37. Case Study 2: Queue Lengths at Station 9.	152
38. Case Study 2: Queue Lengths at Station 10.	153
39. Effect of Mean Interarrival Time on Airfield Throughput.	154
40. Effect of Mean Interarrival Time on Response Time.	154
41. Effect of Mean Interarrival Time on Aircraft on Station.	155
42. Effect of Constrained Resources on Throughput, Analytical Results.	155
43. Effect of Constrained Resources on Response Time, Analytical Results.	155
44. Effect of Constrained Resources on Aircraft on Station, Analytical Results.	156
45. Effect of Constrained Resources on Throughput, Simulation Results.	156
46. Effect of Constrained Resources on Response Time, Simulation Results.	156
47. Effect of Constrained Resources on Aircraft on Station, Simulation Results.	157
48. Effect of Constrained Resources on Throughput, Relative Errors (Percent).	157

Table	Page
49. Effect of Constrained Resources on Response Time, Relative Errors (Percent).	157
50. Effect of Constrained Resources on Aircraft on Station, Relative Errors (Percent).	158
51. Effect of Constrained Resources on Response Time, Deterministic Results.	158
52. Effect of Constrained Resources on Response Time, Relative Errors in Deterministic Results (Percent).	158
53. Effect of Squared Coefficient of Variation on Airfield Throughput. . .	159
54. Effect of Squared Coefficient of Variation on Time on Response Time.	159
55. Effect of Squared Coefficient of Variation on Aircraft on Station. . .	159

Abstract

A single-chain nested fork-join queuing network (FJQN) model of mobility airfield ground processing is proposed. In order to analyze the queuing network model, advances on two fronts are made. First, a general technique for decomposing nested FJQNs with probabilistic forks is proposed, which consists of incorporating feedback loops into the embedded Markov chain of the synchronization station, then using Marie's Method to decompose the network. Numerical studies show this strategy to be effective, with less than two percent relative error in the approximate performance measures in most realistic cases. The second contribution is the identification of a quick, efficient method for solving for the stationary probabilities of the $\lambda(n)/C_k/r/N$ queue. Unpreconditioned Conjugate Gradient Squared is shown to be the method of choice in the context of decomposition using Marie's Method, thus broadening the class of networks where the method is of practical use. The mobility airfield model is analyzed using the strategies described above, and accurate approximations of airfield performance measures are obtained in a fraction of the time needed for a simulation study. The proposed airfield modeling approach is especially effective for quick-look studies and sensitivity analysis.

NESTED FORK-JOIN QUEUING NETWORKS AND THEIR APPLICATION TO MOBILITY AIRFIELD OPERATIONS ANALYSIS

I. Introduction

Motivation

A critical aspect of military air mobility analysis is the study of the relationship between a mobility airfield's resources and the characteristics of the stream of mobility aircraft scheduled to arrive at that airfield. This relationship, which is loosely termed *airfield capability*, can best be understood by answering a series of questions [114]:

1. Where are the bottlenecks in the processing of aircraft on the ground?
2. What levels of critical resources are needed to maintain a given level of throughput¹?
3. Can a base's critical resources support a planned throughput level?
4. What is the airfield's maximum on the ground (MOG)²?
5. What is the airfield's maximum achievable throughput?

In order to answer these questions, the mobility analyst must be able to quantify airfield capability. The following quantities are usually used as capability measures [114]:

¹Airfield throughput is the amount of cargo and the number of passengers that can be moved through the airfield in a given amount of time [129].

²An airfield's MOG is the number of aircraft that can land at that airfield, process through it, and take off again in a predetermined amount of time [83].

1. The throughput capacity (in tons of cargo and number of passengers).
2. Critical resource levels required to sustain a given throughput level.
3. MOG.
4. The probability distribution of the time an aircraft is on the ground.

Over the years, several analyses of airfield capability have been conducted in an attempt to provide a mechanism for answering some or all of these questions. However, none of the models proposed by these studies fully captures the stochastic nature of events at an airfield. To remedy this situation, analysts at the United States Air Force's Air Mobility Command (AMC) initiated the Base Resource and Airfield Capability Evaluation (BRACE) study in 1994. The purpose of the BRACE study is to provide AMC with the simulation modeling capability to directly address the five issues listed above, while explicitly allowing for the variability inherent in airfield operations. The BRACE simulation model enables the estimation of the airfield capability measures listed above. Ultimately, AMC desires to use the output of the BRACE model as input to analyses of system-wide airlift operations [112].

Problem Statement

The processing of aircraft at a mobility airfield requires a complex arrangement of synchronized tasks. AMC's BRACE study focuses on simulating these activities. As a parallel effort, AMC has proposed using a queuing network model to study an airfield's operations. An analytical approach may require some parts of the system to be modeled at lower resolution than that used in the BRACE simulation model; nevertheless, the queuing network model should have a structure that is similar to the network that serves as the basis of the BRACE simulation model, so that comparison between the output of the two models will be meaningful. The analytical model is intended to provide AMC with a first step toward the following capability:

1. An additional mechanism for answering the questions on Page 1, albeit at a more gross level than the BRACE model (as for a quick-look study).
2. A tool for analyzing the sensitivity of mobility operations at an isolated airfield.
3. A tool for reducing the variance in the BRACE simulation model runs.

The Analytical Airfield Model

Overview. In this section, a queuing network model of mobility airfield operations is proposed and described in detail. This model, which will be referred to as the Analytical Airfield Model (AAM), is based on the BRACE simulation implementation, which is discussed in detail in References [107], [114], and [115].

The Network Structure. The general task precedence graph for the ground flow of an aircraft at a mobility airfield is shown in Figure 1. In this graph, each box represents a task that may have to be performed while an aircraft visits the air base³. The flow of precedence among tasks is from top to bottom; that is, tasks at the top of the graph must be completed before the tasks below them. Tasks that typically begin at the same time are immediately preceded by an appropriately labeled horizontal bar. "Refuel" and "concurrent maintenance" are examples of tasks that begin simultaneously. If a bar labeled "synchronize" immediately follows two or more tasks, those tasks must be complete before any task below the bar may begin. For example, the "liquid oxygen servicing" and "cargo on" tasks must be complete before the "bags/passengers on" task can begin.

We can construct a queuing network that is logically equivalent to the flow of tasks by viewing the service providers as "stations," and the aircraft as "customers" that move through the network visiting the stations. The graph of the equivalent

³Note that this list of tasks is intended to be as general as possible. The assumptions made in a particular mobility study may require ignoring one or more of these tasks (e.g., if a particular contingency plan has no passenger requirements, the passenger onload and offload tasks would be ignored).

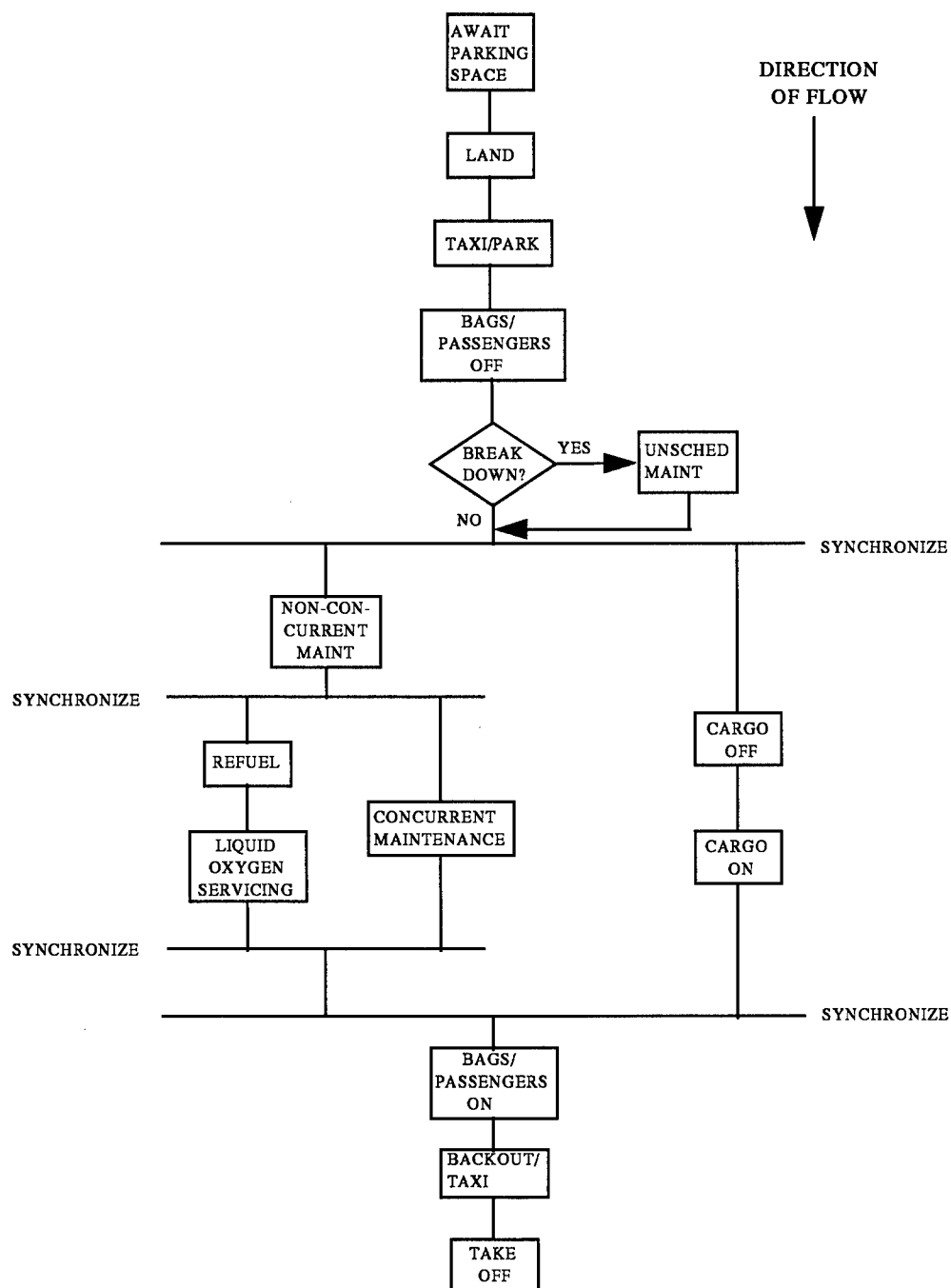


Figure 1. Task Precedence Graph.

Table 1. Network Station Descriptions.

Station Number	Activity Description	Number of Servers	Service Discipline	Distribution Type	Visit Prob
1	Landing	$< N$	FCFS	Phase Type	1
2	Taxi/Park	N	Delay	Phase Type	1
3	Passengers/bags off	N	Delay	General	< 1
4	Unscheduled Maintenance	$< N$	FCFS	Phase Type	< 1
5	Scheduled Maintenance (not concurrent with refueling)	N	Delay	General	< 1
6	Refuel	$< N$	FCFS	Phase Type	< 1
7	Liquid oxygen servicing	N	Delay	General	< 1
8	Scheduled Maintenance (concurrent with refueling)	N	Delay	General	< 1
9	Cargo off/on	$< N$	FCFS	Phase Type	< 1
10	Passengers/bags on	N	Delay	General	< 1
11	Backout/Taxi	N	Delay	General	1
12	Takeoff	$< N$	FCFS	Phase Type	1
13	Standard Ground Delay	N	Delay	Deterministic	1

Note: $N \equiv$ network capacity; FCFS \equiv first-come-first-served.

queuing network is shown in Figure 2. A description of the network stations is in Table 1; this information is discussed in detail below.

The diamond-shaped nodes in Figure 2 are *fork nodes* (labeled “F”) and *join nodes* (labeled “J”). These nodes provide a convenient way to model activities that must be synchronized. To illustrate the function of the fork and join nodes, we consider an aircraft that needs to be refueled, and that has maintenance scheduled to be performed at the same time as the refueling. It must therefore visit Nodes 6 and 8 simultaneously. The aircraft is said to “fork” because it behaves as if it has split into two linked “clones;” one of these clones visits Node 6 (and perhaps Node 7) while the other clone is at Node 8. The clone that finishes service first awaits the other clone at the join node. When the second clone finishes service, the two clones immediately “join” into a single unit and proceed through the remaining tasks.

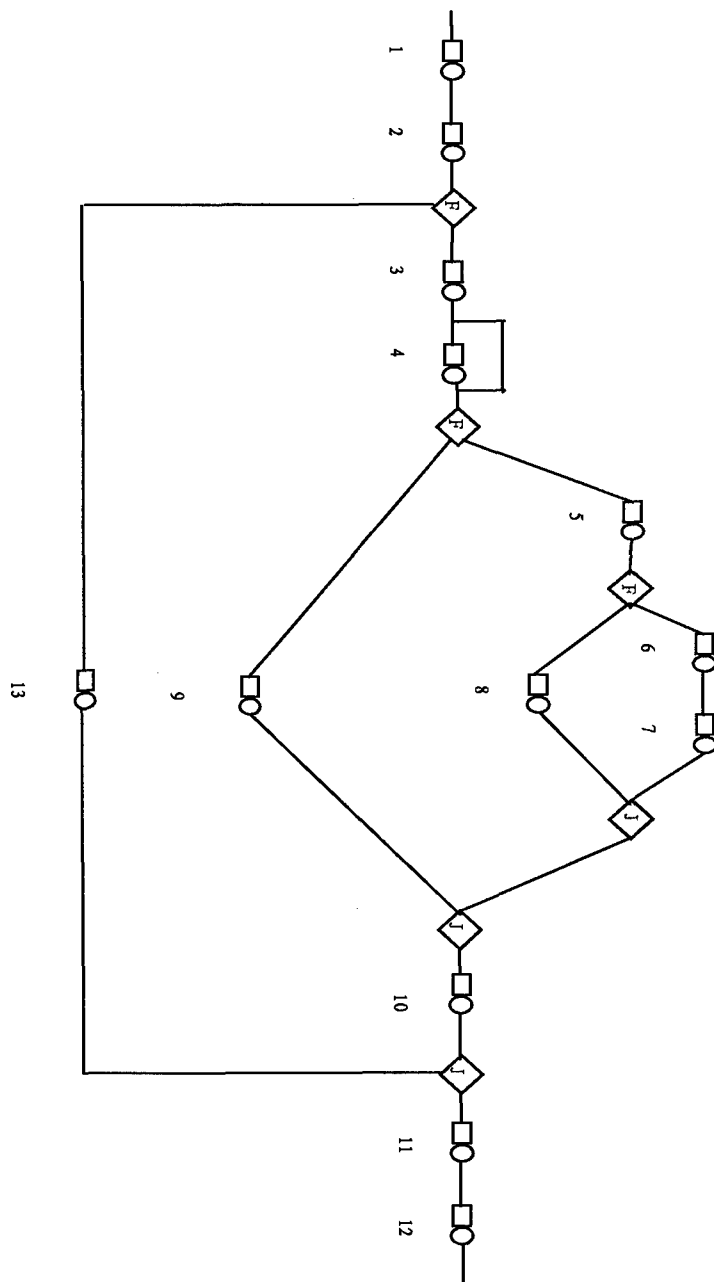


Figure 2. Graph of the Equivalent Queueing Network.

The probability that an aircraft visits a particular service station (or a subnetwork of stations) is assumed to be independent of the number and type of aircraft at the stations in the network. This is a realistic assumption, since the types of processing required by a particular aircraft are a function of the flight schedule, the aircraft manifest or en-route events, and are not linked explicitly to any flight line activities [114].

For the purposes of this study, it will be assumed that flow through the queuing network reaches steady state. During wartime or a contingency, mobility operations are normally conducted around the clock over a period of weeks or months. Therefore, it is reasonable to assume that, after some initial warm-up period, the mean aircraft flow rate remains relatively steady over time. For this reason, AMC's airfield simulations will be primarily non-terminating and will focus on steady-state results [112]. Thus, it is realistic to use a steady-state queuing network model.

Another assumption that is implied by the model structure is that crew rest requirements have no impact on aircraft delay; that is, it is assumed that there are enough crews in the system that a crew in need of rest can be immediately replaced by a fresh crew. Since it is common during wartime or contingency operations to preplace crews at different locations throughout the airlift system, it is reasonable to ignore the impact of crew rest [113].

Customer Classes. Normally, a mobility airfield must be able to accept different types of aircraft. This is necessary because the different types of aircraft have different service time distributions at certain network stations, and because the routing probabilities within the network differ for each type. In the AAM, however, differences between aircraft classes will not be modeled explicitly in the interest of model simplicity. However, the differences can easily be accounted for implicitly through the judicious selection of routing probabilities and the composition of service and arrival laws.

Network Capacity. Since an airfield obviously has limited parking space, a finite capacity must be imposed on the queuing network model. The number of aircraft permitted at each station is limited only by the system capacity, since it is possible (but not likely) that all aircraft at the base could be either awaiting or participating in the same kind of activity.

The Arrival Processes. The arrival process is assumed to be a phase-type renewal process (perhaps with a load-dependent mean arrival rate). In actuality, the arrival process of aircraft to an airfield may not even be a renewal process. However, little accuracy is lost by imposing the arrival assumption if the distribution of time between events in a generic point process is approximated by a phase-type distribution whose first few moments match those of the original inter-event distribution [131].

The Service Stations. As stated above, Table 1 provides a description of the number of servers, service discipline, service time distribution, and visit probability at each network station. The service time distribution parameters are not provided since they depend on the airfield being studied. Some stations may have load-dependent service rates.

The passenger/bag loading processes (Stations 4 and 12), liquid oxygen service (Station 8), and the maintenance processes (Stations 7 and 9) are modeled as delay stations. Consultations with logistics experts throughout AMC indicate it is reasonable to view these resources as unlimited [113]. The infinite-server model is appropriate because the service rates do not depend on the number at the station.

The remaining processes are modeled either as delay stations or as multiserver, first-come-first-served (FCFS) queues with phase-type service distributions. As with the arrival process, the assumption of a phase-type form is made for tractability.

Research Goals

In order to meet the needs described in the problem statement, this research will attempt to meet the following objectives:

1. Develop a method for analyzing queuing networks of the same class as the network used in the AAM.
2. Demonstrate the use of the AAM by modeling operations under a particular contingency scenario at a typical mobility airfield, using the method developed under Objective 1 to analyze the AAM network.

Sequence of Presentation

The remainder of this dissertation is divided into six chapters. Chapters II and IV provide theoretical background, and pose problems that need to be resolved in order to satisfy the first research objective. The necessary theoretical advances are presented in Chapters III and V, respectively. Chapter VI, which contains a demonstration of the AAM, satisfies the second research objective. The dissertation concludes with Chapter VII, a summary of contributions and research recommendations.

II. Analyzing Queuing Networks

Overview

This chapter has two purposes. First, it provides motivation for the computational study of queuing network analysis techniques in Chapter III. In addition, it discusses fundamental queuing network theory that the reader needs to understand the material on fork-join queuing networks presented in Chapters IV and V.

The remainder of this chapter assumes a familiarity with queuing theory on the level presented in Reference [59]. The next section briefly covers foundational material. The chapter continues with a detailed discussion of product-form approximations, which are referred to extensively in Chapters IV and V. Computational issues raised by applying product-form approximations to networks with multiserver stations are discussed in the final section.

Basic Concepts

A queuing network is simply a set of queues, together with a set of arcs, that form a directed graph. In a queuing network model, one typically defines system behavior in terms of activity at the nodes (also called stations or member queues) rather than on the arcs. This differs from other network models such as transportation models, in which the emphasis is on flow through the arcs.

Queuing networks may be divided into two broad classes: open and closed. Customers enter and depart an *open* network at one or more points; if the number of customers in an open network is limited, the network is referred to as *capacitated*. In contrast, a *closed* queuing network contains a fixed number of customers that start in the system and remain there, circulating among the network stations. It may be shown that, under certain broad conditions, a capacitated open network has an equivalent closed network representation [38].

A queuing network may be composed of a linked set of graphs, each for a different type of customer. In this case, the network is called a *multiple-chain* or *multichain* network, and a subnetwork visited by a particular customer class is called a *chain*. Networks with only one class of customer are called *single-chain* networks.

Suppose we have a k -station single-chain queuing network of general topology. Define a state of the network by $\tilde{n} \equiv \{n_1, \dots, n_k\}$, where n_i is the number of customers at station i . Suppose the probability that the system is in a particular state \tilde{n} equals the product of the marginal probabilities that there are n_i customers at station i , $i = 1, \dots, k$:

$$\Pr[\tilde{n}] = \prod_{i=1}^k \Pr_i[n_i] \quad (1)$$

If this condition holds, the network is said to be a *separable* or *product-form* network (this definition can easily be generalized for multiple-chain networks). Such networks are usually easy to analyze. In particular, if a separable network is either closed or capacitated, its performance measures can be determined in a straightforward manner using techniques such as convolution [29] and mean value analysis [100].

Unfortunately, most queuing networks that arise in real applications are not separable. While exact performance measures can sometimes be obtained for some of these networks, in many practical cases exact analysis is intractable. Therefore, it is common to seek an approximate solution using more straightforward methods. One such approach is *decomposition*. The goal of decomposition is to approximate the network's performance measures by partitioning it into a set of subnetworks, which are then analyzed in isolation [68], [70]. The next section describes in detail a class of decomposition techniques that attempt to take advantage of the ease with which separable networks can be analyzed.

Product-Form Approximation of Closed Queuing Networks

Background. Product-form approximation is a type of decomposition that works well for closed networks. In this technique, the performance measures of the original network are approximated by those of a closed, product-form network of similar characteristics and behavior. In a product-form approximation, the original network is partitioned into a set of subnetworks, which are analyzed in isolation (that is, as independent networks) to get approximate throughput levels that are conditioned on subnetwork population. For each subnetwork, an associated exponential server with load-dependent service rates is constructed; these service rates are set equal to the conditional throughput levels of the original subnetwork. The throughput levels are calculated in such a way that flow into and out of the exponential station closely approximates the flow behavior of the original subnetwork. A separable network is then formulated by replacing the subnetworks in the original network topology by the flow-equivalent servers; the performance measures of this network are used as approximations of those of the original network. Figure 3 illustrates the process of server replacement. Clearly, the error in such an approximation comes from two sources: the assumption of exponential service, and the approximation of the conditional throughputs through isolated analysis [23].

Building on the work of Dallery and Cao [39], Baynat and Dallery identify four conditions a network partition must meet in order for a product-form approximation to be reasonably accurate [23]:

1. Customers enter and leave subnetworks one at a time.
2. The state-dependent behavior of a subnetwork is independent of the behavior of its complement.
3. The routing between a subnetwork and its complement is independent of the state of the subnetwork.
4. The splitting and matching of customers occurs only within subnetworks.

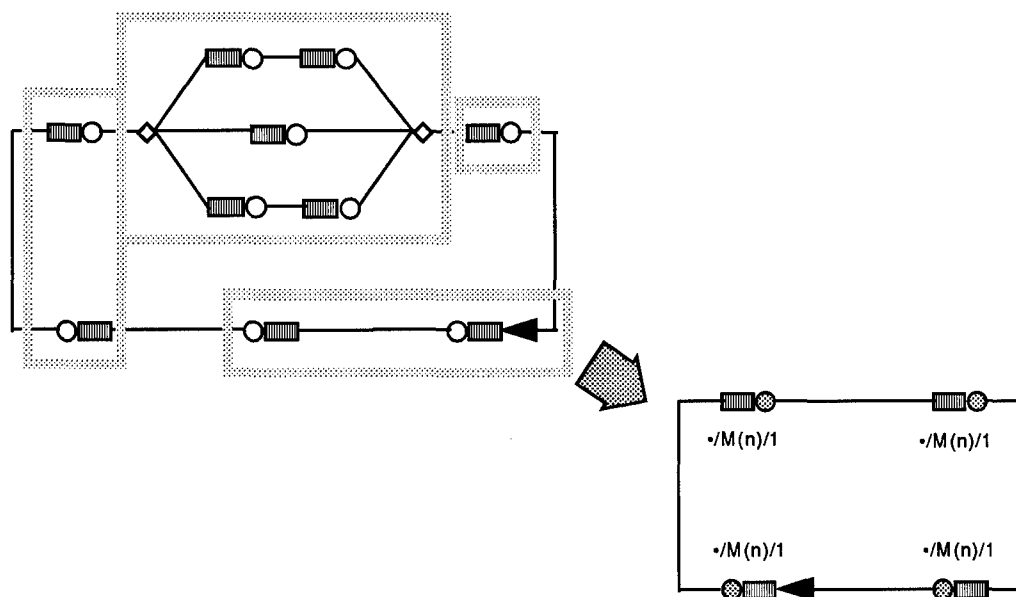


Figure 3. The Product-Form Approximation Technique.

Baynat and Dallery call a partition that meets these conditions *feasible*.

Given a feasible partition, the estimation of the conditional throughputs becomes the central issue in a product-form approximation. The two most common techniques for computing these throughputs are *aggregation* and *Marie's Method*. A discussion of each approach follows.

Aggregation. Aggregation has its roots in the work of Avi-Itzhak and Heyman [5] and Chandy, Hertzog, and Woo [32], [33], and is motivated and described fully in [72:161ff]. In this method, the subsystem to be isolated is analyzed as a closed, independent network. This network is formed by "short-circuiting" the subnetwork's complement (that is, removing the complement from the network). Approximate conditional throughput levels are obtained by calculating the throughput of this new subnetwork for fixed population levels. Chandy, Hertzog, and Woo showed

that these conditional throughputs are exact if the original network is separable [32]. In the case where the network is nearly completely decomposable (that is, the behavior of the subnetworks is nearly mutually independent), the error induced by using aggregation will be small [23].

Marie's Method. Marie's Method was proposed by Raymond Marie in 1979 [78], [81]. The central idea of the method is to analyze the subsystem of interest as an isolated, open network with finite capacity and load-dependent Poisson arrivals.¹ The load-dependent throughput levels of the isolated subsystem become the load-dependent mean service rates of the associated exponential server.

Before describing Marie's Method for single-chain networks, we define the symbols used. Let N be the number of customers in the original closed network. Let n_i be the number of customers in subsystem i ($0 \leq n_i \leq N$). Further, define $\lambda_i(n_i)$ and $\tilde{\nu}_i(n_i)$ as the arrival rate and the conditional throughput of customers at subsystem i , $n_i = 0, \dots, N-1$. Let $\mu_i(n_i)$ be the load-dependent service rate for the associated flow-equivalent exponential server, $n_i = 0, \dots, N$.

Marie's Method makes use of a system of three sets of foundational equations. The first equation set, which is derived by applying the Marginal Local Balance Theorem [59:231-232], establishes the throughput levels of the isolated subnetwork:

$$\tilde{\nu}_i(n_i) = \lambda_i(n_i - 1) \frac{\tilde{P}_i(n_i - 1)}{\tilde{P}_i(n_i)}, \quad n_i = 1, \dots, N \quad (2)$$

The probabilities $\tilde{P}_i(n_i)$ are the marginal probabilities that n_i customers are found in subsystem i ; these are found by analyzing the subsystem in isolation as described above. We get the load-dependent service rates of the flow-equivalent exponential

¹Where the subsystem consists of a single station, this reduces to analyzing an isolated $\lambda(n)/G/c/N$ queue. When the service time distribution can be modeled as a Coxian distribution [37], efficient algorithms exist for calculating the performance measures of the isolated queue when $c = 1$ [79], [80]. The case where $c > 1$ is considered in [117] and in Chapter III.

server by setting them equal to the throughput levels of the isolated subsystem:

$$\mu_i(n_i) = \tilde{\nu}_i(n_i), \quad n_i = 1, \dots, N \quad (3)$$

The final equation set, which is also derived using the Marginal Local Balance Theorem, ensures local balance in the approximate product-form network:

$$\lambda_i(n_i) = \mu_i(n_i + 1) \frac{\hat{P}_i(n_i + 1)}{\hat{P}_i(n_i)}, \quad n_i = 0, \dots, N - 1 \quad (4)$$

The probabilities $\hat{P}_i(n_i)$ are derived by analyzing the associated product-form network with any appropriate technique.

Marie's algorithm solves Equations (2), (3), and (4) for the conditional throughputs using fixed-point iteration. The algorithm is given below:

STEP 0. Initialize $\mu_i(n_i)$ for $n_i = 1, \dots, N$.

STEP 1. Calculate $\lambda_i(n_i)$ using Equation (4).

STEP 2. Analyze the station in isolation to get $\tilde{P}_i(n_i)$, $n_i = 0, \dots, N$.

STEP 3. Use Equation (2) to get $\tilde{\nu}_i(n_i)$, $n_i = 1, \dots, N$.

STEP 4. Calculate the load-dependent service rates $\mu_i(n_i)$ for the replacement server using Equation (3).

STEP 5. Repeat Steps 1 through 4 until the relative improvement in the $\mu_i(n_i)$ values is less than some specified tolerance value.

The measure of improvement normally used is the maximum relative change in the elements of the service rate vector:

$$\max_{i, n_i} \left| \frac{\mu_i^{(k)}(n_i) - \mu_i^{(k-1)}(n_i)}{\mu_i^{(k-1)}(n_i)} \right| < \varepsilon \quad (5)$$

where $i = 1, \dots, k$, $n_i = 0, \dots, N \forall i$, and ε is the selected tolerance (typically set at 10^{-3} or 10^{-4}).

Marie's Method appears in several different studies as an accurate technique for analyzing non-separable networks [20], [22], [24], [27], [38], [103]. Marie's method compares favorably to aggregation, and provides superior estimates of expected queue lengths in many cases [23]. Bondi and Whitt find that Marie's Method is the most accurate and stable of the decomposition techniques they examine, although they suggest further study of its convergence properties and robustness [27].

Baynat and Dallery have extended Marie's Method so it can be used to analyze closed networks with $R (> 1)$ chains. Their approach is based on decomposing the network chain by chain. The derivation is similar to that for the single-chain case, although each equation set must now be generated for each of the R chains. With the obvious extensions to the notation, the multiple-chain analogs to Equations (2), (3), and (4) are:

$$\tilde{\nu}_{ri}(n_{ri}) = \lambda_{ri}(n_{ri} - 1) \frac{\tilde{P}_{ri}(n_{ri} - 1)}{\tilde{P}_{ri}(n_{ri})}, \quad n_{ri} = 1, \dots, N_r, \quad r = 1, \dots, R \quad (6)$$

$$\mu_{ri}(n_{ri}) = \tilde{\nu}_{ri}(n_{ri}), \quad n_{ri} = 1, \dots, N_r, \quad r = 1, \dots, R \quad (7)$$

$$\lambda_{ri}(n_{ri}) = \mu_{ri}(n_{ri} + 1) \frac{\hat{P}_{ri}(n_{ri} + 1)}{\hat{P}_{ri}(n_{ri})}, \quad n_{ri} = 0, \dots, N_r - 1, \quad r = 1, \dots, R \quad (8)$$

The multiple-chain version of Marie's algorithm is as follows [20], [26]:

STEP 0. Initialize $\mu_{ri}(n_{ri})$ for $r = 1, \dots, R$ and $n_{ri} = 1, \dots, N_r$.

STEP 1. For $r = 1, \dots, R$, calculate $\lambda_{ri}(n_{ri})$ using Equation (4).

STEP 2. Analyze the station in isolation to get $\tilde{P}_{ri}(n_{ri})$, $r = 1, \dots, R$ and $n_{ri} = 0, \dots, N_r$.

STEP 3. Use Equation (2) to get $\tilde{\nu}_{ri}(n_{ri})$, $r = 1, \dots, R$ and $n_{ri} = 1, \dots, N_r$.

STEP 4. Calculate the load-dependent service rates $\mu_{ri}(n_{ri})$ for the r th replacement server using Equation (3).

STEP 5. Repeat Steps 1 through 4 until the relative improvement in the $\mu_{ri}(n_{ri})$ values is less than some specified tolerance value.

The stopping test is similar to that given in Equation (5), except that the maximization is also performed over the R chains in the network:

$$\max_{i,n_i,r} \left| \frac{\mu_{ir}^{(k)}(n_{ir}) - \mu_{ir}^{(k-1)}(n_{ir})}{\mu_{ir}^{(k-1)}(n_{ir})} \right| < \varepsilon \quad (9)$$

where $i = 1, \dots, k$, $n_i = 0, \dots, N \forall i$, $r = 1, \dots, R$, and ε is the selected tolerance.

As with the single-chain version, the analysis of subsystems in isolation drives the time complexity of Marie's Method. The multiple-chain analysis is further complicated if a subsystem to be analyzed is visited by more than one customer class. If the number of chains a subsystem belongs to is large, the analysis in isolation will not be practical. Baynat and Dallery have proposed a class aggregation technique to deal with this difficulty for the case where the subsystem is a single queue; this simplified approach is approximate, but appears to induce minimal additional error [20].

Analyzing Multiserver Stations Using Marie's Method

In much of the open literature, practical implementation of Marie's Method is restricted to queuing networks composed solely of single-server FCFS stations with k -Coxian service laws. This is likely due to two factors:

1. A general service time distribution can be approximated to an arbitrary degree of closeness by matching its moments to those of a k -Coxian law (the Coxian representation is exact when the original distribution has a rational Laplace transform) [37].
2. Isolating the station and approximating the service law by a Coxian distribution yields a $\lambda(n)/C_k/1/N$ queue; efficient methods exist for analyzing this type of station.

Marie proposes an algorithm for determining the stationary queue length probabilities of $\lambda(n)/C_k/1/N$ queue, with specialized variants for the case where the service time distribution is k -Erlang or k -hyperexponential [79], [81]. Later refinements to Marie's algorithms incorporate the possibility of feedback to the queue [80].

Queues with multiple servers have received far less attention in the literature. It is possible, however, to analyze a general multiserver station in isolation. Based on the above reasoning, this requires the formulation of a $\lambda(n)/C_k/r/N$ queue, and subsequent analysis of this queue's embedded continuous-time Markov chain (CTMC) (see Appendix A for a discussion of CMTC analysis). Stewart and Marie propose a method for calculating stationary probabilities for this chain when the servers are homogeneous; their approach is described in detail in Chapter III. In addition to giving an efficient method for formulating the transition rate matrix Q for this Markov process, Stewart and Marie suggest calculating the stationary probability vector π by using simultaneous iteration to solve for the appropriate eigenvector of the transition probability matrix P . Their numerical experiments showed that, based on the computing power available at that time (the late 1970s), time complexity made implementation impractical for chains with more than 1,400 states [117].

It may be because of this 1,400-state limitation that later researchers (such as Baynat and Dallery) have limited their study of Marie's method to the case where $r = 1$. Unfortunately, many multiserver decomposition problems have embedded CTMCs with dimension much greater than 1,400. In fact, it is not hard to find problems with upwards of 50,000 states, as we will see in Chapter III. The stationary probability problem of the $\lambda(n)/C_k/r/N$ queue needs to be solved quickly and efficiently for large-dimension systems if multiserver decomposition with Marie's method is to be practical. Since Stewart and Marie's article was published, many advances have been made in computational linear algebra which can be used to speed solution of large stationary probability problems; in addition, computing power has increased dramatically since the early 1980s. The goal of the next chapter is to exploit these

advances to increase the size of the embedded chain for which numerical solution is practical, thereby widening the range of queuing networks that can be easily analyzed using Marie's method.

III. Efficient Isolated Analysis of General Multiserver Stations During Decomposition of Closed Queuing Networks

Introduction and Motivation

Suppose we have a closed queuing network that is not of product form, in part because it contains a station with first-come-first-served discipline and r servers with identical general service time distributions. If we decide to use Marie's decomposition method to get approximate performance measures for the network, we need a strategy for analyzing the station of interest in isolation. Assuming that we can reformulate this station's service law in k -Coxian form (as discussed in Chapter II), we can analyze it as an isolated $\lambda(n)/C_k/r/N$ queue. By exploiting the Markovian structure of this queue's stationary behavior, we can derive its stationary probability distribution, which can then be passed back to Marie's algorithm. For this approach to be successful, we need a fast, efficient method for calculating the stationary probability distribution of the isolated queue.

The purpose of this chapter is to exploit recent advances in computational probability to identify a fast, efficient method for finding these probabilities, thereby making the application of Marie's method practical for a much larger class of decomposition problems. The discussion begins with a description of Stewart and Marie's method for generating the transition rate matrix Q of the embedded CTMC¹, and an analysis of the matrix's structure. The insights gained from studying Q are used to develop a list of candidate solution methods, each of which are used to calculate the stationary probability vector π for a set of 46 representative large-dimension problems. Based on this computational experience, a preferred solution approach is identified, and its effectiveness is demonstrated by analyzing the performance of four

¹The discussion in this chapter assumes the reader is familiar with the general concepts presented in Appendix A. All acronyms, abbreviations, and concepts introduced here are defined in that appendix.

sample queuing networks. The chapter closes with a statement of the conclusions drawn from this research.

The Transition Rate Matrix for the $\lambda(n)/C_k/r/N$ Queue

Constructing the Matrix. This section summarizes Stewart and Marie's method for constructing the transition rate matrix Q of the embedded CTMC of a $\lambda(n)/C_k/r/N$ queue [117]. We begin by defining a state of the process as the $(k+1)$ -tuple (n, h_1, \dots, h_k) , where n is the number of customers in the queue, and h_i is the number of these customers in phase i of service. Stewart and Marie show that m , the total number of states (the order of Q) is

$$m = \left[\sum_{i=0}^r \binom{i+k-1}{i} \right] + (N-r) \binom{r+k-1}{r} \quad (10)$$

Figure 4 shows the strong effect of k and r on m , and how the state space dimension can become very large very quickly, especially for $k \geq 4$ and $r \geq 10$.

Stewart and Marie's algorithm for generating Q requires that the states be ordered a certain way. For a given value of n , the states must be sorted first in descending order by h_1 , then in descending order by h_2 , and so on through h_k . Table 2 shows a sample ordering for $k = 4$, $r = 3$, and $3 < n < N$.

Once we have properly ordered the states, we can efficiently construct Q row by row. Define s as the number of the state corresponding to the row of Q under construction. Suppose also that $\alpha = \{\alpha_i; i = 1, \dots, k-1\}$ is the vector of probabilities that service will continue past phase i , and that $\mu = \{\mu_i; i = 1, \dots, k\}$ is the vector of service rates for the stages of Coxian service. For each source state s , transition can take place to a destination state d due to one of three types of events: an arrival, a transition between service phases, or a departure from the system. State transitions due to arrivals take place at rate $\lambda(n)$; the appropriate entry

Table 2. Ordering of States When $k = 4$, $r = 3$, and $3 < n < N$.

n	h_1	h_2	h_3	h_4
\vdots	\vdots	\vdots	\vdots	\vdots
n-1	0	0	1	0
n-1	0	0	0	0
n	3	0	0	0
n	2	1	0	0
n	2	0	1	0
n	2	0	0	1
n	1	2	0	0
n	1	1	1	0
n	1	1	0	1
n	1	0	2	0
n	1	0	1	1
n	1	0	0	2
n	0	3	0	0
n	0	2	1	0
n	0	2	0	1
n	0	1	2	0
n	0	1	1	1
n	0	1	0	2
n	0	0	3	0
n	0	0	2	1
n	0	0	1	2
n	0	0	0	3
n+1	3	0	0	0
n+1	2	1	0	0
\vdots	\vdots	\vdots	\vdots	\vdots

Adapted from Reference [117].

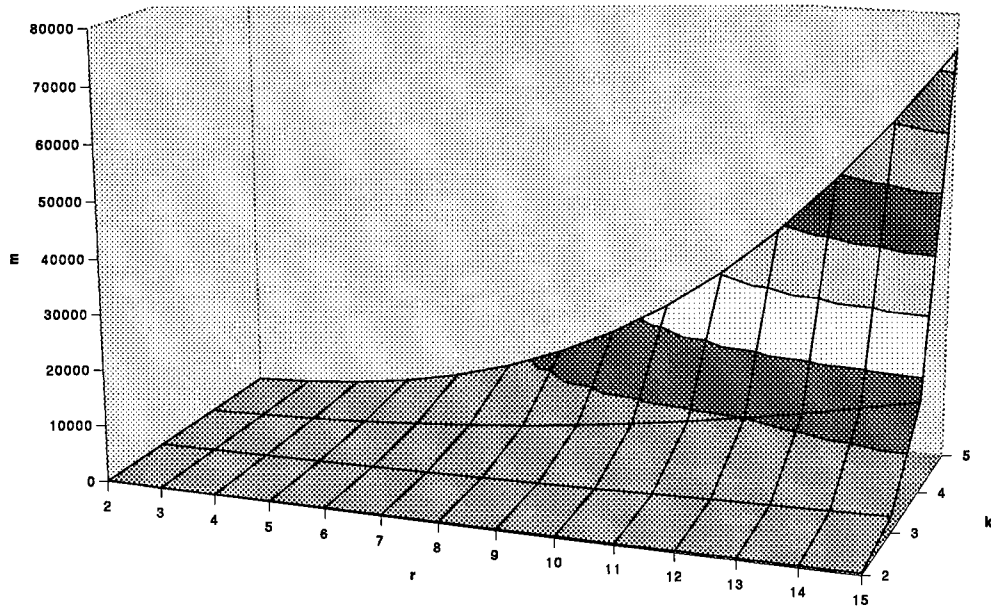


Figure 4. Surface Contour Plot of m versus k and r , $N = 30$.

in Q is $q_{s,d} = \lambda(n)$, where

$$d = s + \binom{n+k-1}{n} \quad (11)$$

If $h_i \neq 0$, a transition from phase i to phase $i+1$ of service takes place at rate $\alpha_i \mu_i$.

Thus, $q_{s,d(i)} = \alpha_i \mu_i$, where

$$d(i) = \begin{cases} s + \binom{n - [\sum_{j=1}^i h_j] + k - i - 1}{n - [\sum_{j=1}^i h_j]} & n \leq r \\ s + \binom{r - [\sum_{j=1}^i h_j] + k - i - 1}{r - [\sum_{j=1}^i h_j]} & n > r \end{cases} \quad (12)$$

and $i = 1, \dots, k-1$. Entries in the s th row of Q that are due to departures are given by $q_{s,d(i)} = (1 - \alpha_i)\mu_i$, where $h_i \neq 0$, and

$$d(i) = \begin{cases} s - \sum_{j=0}^{i-1} \binom{n - [\sum_{l=1}^j h_l] + k - j - 2}{n - [\sum_{l=1}^j h_l] - 1} & n \leq r \\ s - \binom{r + k - 1}{r} & n > r, i = 1 \\ s - \binom{r + k - 1}{r} - \sum_{j=0}^{i-1} \binom{n - [\sum_{l=1}^j h_l] + k - j - 2}{n - [\sum_{l=1}^j h_l] - 1} & n > r, i > 1 \end{cases} \quad (13)$$

and $i = 1, \dots, k-1$ (note that $d(k) = \mu_k$, provided $h_k \neq 0$). For detailed derivations of Equations (11), (12), and (13), see Reference [117]. As is usual for a CTMC, the diagonal elements of Q are formed using the relation

$$q_{ss} = - \sum_{d \neq s} q_{sd} \quad (14)$$

The Structure of the Matrix. Stewart and Marie show that the generation scheme described above results in a Q matrix of order m that has the block structure shown in Figure 5. In that figure, the block marked Q_0 has order $m_1 = \sum_{i=0}^r \binom{i+k-1}{i}$ and is referred to as the "initialization section" of the matrix. The blocks marked S , D , and T are each of order $m_2 = \binom{r+k-1}{r}$. The S blocks contain entries of Q corresponding to the departure process; all of these blocks are identical. The D blocks contain the matrix diagonal entries and the interphase transition rates; these blocks vary only in the diagonal elements q_{ss} , since the transition rates between service phases remain constant from state to state. The T blocks contain the arrival rates, and are of the form $\lambda(n)I$. All entries outside the marked blocks are zero.

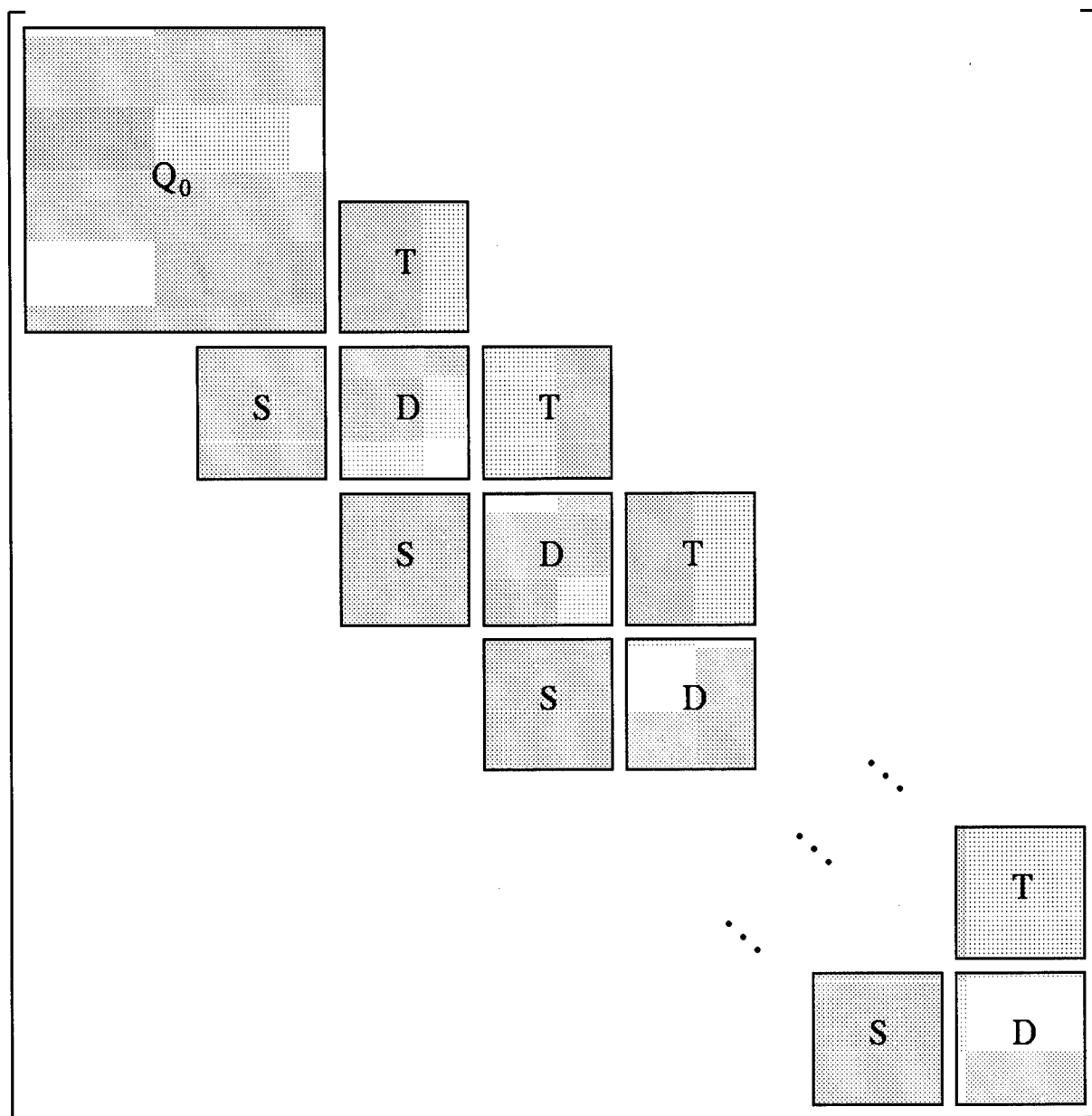


Figure 5. Block Structure of Q .

Solution Methods: Narrowing the Choice

Appendix A describes a wide range of methods for deriving π for a CTMC. By considering these available methods in light of the structure of Q and the context of our problem, we can narrow our choice to a subset of techniques that seem promising.

Direct methods would be the best choice for a small number of states. However, memory constraints limit their applicability to 2,500 states or less, even for sparse matrix implementations such as GE. Unfortunately, this is not much of an increase in capability over the limitation encountered by Stewart and Marie.

If Q were NCD, we could make a good case for the use of decomposition. However, an inspection of Figure 5 shows no clear way to partition Q so that it is decomposable. Wherever one attempts a partition, one must deal with the effect of ignoring S and T blocks. Since the norms of these blocks are known to be similar in order to those of the D blocks, a decomposition of Q would likely induce substantial error.

We might try a block-recursive approach to solving for the stationary probabilities. Unfortunately, Q cannot be put in a suitable block-Hessenberg form unless m_2 evenly divides m_1 (in which case Q_0 can be partitioned into submatrices of order m_2); clearly, this condition does not uniformly hold. Since most block-recursive solution methods require that Q be partitioned into submatrices of equal order, their usefulness is limited in this problem domain.

We are left, then, with iterative solution methods. As shown in Appendix A, many iterative algorithms perform well when used to analyze large-scale CTMCs. The fact that the chain of interest is not NCD considerably lessens the danger of ill conditioning (although the absence of near-complete decomposability is not sufficient to declare the problem well-conditioned); this, in turn, bodes well for the success of an iterative solution approach.

Computational Experience

The Candidate Iterative Methods. Because of the irregular block structure of Q , experimentation was limited to those non-stationary iterative methods that can accomodate nonsymmetric matrices. Representatives of the three classes of methods described in Appendix A were investigated: LSQR, GMRES(j), and CGS. For GMRES, the restart period j was set at 50, 75, and 100 iterations.

In order to assess the the cost tradeoff of preconditioning, each method was used both with and without preconditioners. ILU(0), modified ILU(0), and ILUTH (with tolerance 10^{-3}) were applied, since the necessary matrix decompositions are known to exist for Q^T .

The Representative Queuing Systems. Each of the above iterative methods was used to solve for π for 46 different $\lambda(n)/C_k/r/N$ queues. The parameters N , r , and k were chosen so that the state space of each system was large enough to require the advantages of as iterative solution method ($> 1,000$), yet small enough to enable solution with existing computer resources ($< 50,000$). The actual parameter values, together with the order of each system, are given in Table 3.

For Systems 1 to 23, the service law chosen was k -Erlang with mean service rate one and $\sigma^2 = 1/k$. Systems 24 to 46 used generalized k -Erlang distributions with the parameters given in Table 4.

No service distributions with a coefficient of variation greater than one were considered. These laws can be modeled to arbitrary accuracy by by matching their first few moments to those of a 2-Coxian distribution [121:360-361]. A 2-Coxian law yields a Q matrix of order less than 1,100 over the ranges of N and r used here; such systems can easily be solved directly in minimal time.

Two arrival distributions were considered. Arrival Process I was based on throughput from a product-form queuing network of three stations in series, one of which was a $\cdot/M/r$ queue with unit mean service rates. This arrival rate vector is

Table 3. Representative System Configurations.

Number	N	r	k	$Order$
1/24	45	10	3	2596
2/25	60	10	3	3586
3/26	30	15	3	2856
4/27	45	15	3	4896
5/28	60	15	3	6936
6/29	30	20	3	4081
7/30	45	20	3	7546
8/31	60	20	3	11011
9/32	60	5	4	3206
10/33	15	10	4	2431
11/34	30	10	4	6721
12/35	45	10	4	11011
13/36	60	10	4	15301
14/37	30	15	4	16116
15/38	45	15	4	28356
16/39	60	15	4	40596
17/40	30	20	4	28336
18/41	30	5	5	3402
19/42	45	5	5	5292
20/43	60	5	5	7182
21/44	15	10	5	8008
22/45	30	10	5	23023
23/46	45	10	5	38038

Table 4. Parameters of the Generalized k -Erlang Distributions.

k	μ	μ	σ^2
3	1.0	(6.0, 3.0, 2.0)	0.3889
4	1.0	(20.0, 5.0, 4.0, 2.0)	0.3550
5	1.0	(40.0, 10.0, 8.0, 4.0, 2.0)	0.33875

a reasonable example of the approximate flows that would be encountered during queuing network decomposition. Arrival Process II, in which the arrival rates vary much more widely, is a pathological example of network flow, and was chosen for the purpose of sensitivity analysis; it is defined by the linear relation

$$\lambda(n) = \frac{r(N - n)}{N - r} \quad (15)$$

Computing Environment. The solution and preconditioning routines used in this study are implemented in Version 1.0 of the University of Texas' Fortran 77 subroutine NSPCG [93]. All other code was written in Fortran 90. Compilation and execution took place on a Digital Equipment Corporation (DEC) Alpha workstation. DEC's F90 compiler was used to compile both the Fortran 77 and the Fortran 90 code, which was optimized for speed (DEC Level 4). Eight-bit real and integer data structures were used to enhance precision and provide a worst-case memory usage scenario.

Numerical Analytical Considerations. The initial solution was arbitrarily set at $\pi_j^{(0)} = 1 - (j/m), j = 1, \dots, m$; this $\pi^{(0)}$ was simpler to obtain than the one suggested by Stewart and Marie in [117]. The upper limit on the dimension of the Krylov subspace was set at ten, because this seemed to work well for Philippe et al. [95].

The stopping test used was based on the residual norm $\|Q^T \pi^{(i)}\|_2$. A solution was judged to have converged if $\|Q^T \pi^{(i)}\|_2 \leq 10^{-4}$ within 2,000 iterations.² Stewart and Marie suggest using the tolerance level 10^{-4} based on their observation that it is typically the best that can be expected from Marie's method; they see no point in holding probabilities obtained from isolated analysis to a tighter standard [117].

²This relatively high number was selected to ensure convergence in as many cases as possible.

Execution was terminated without convergence under two conditions: (1) the maximum iteration count of 2,000 was reached, but $\|Q^T \pi^{(i)}\|_2 > 10^{-4}$, and (2) the process stalled (that is, $\|Q^T \pi^{(i)}\|_2$ failed to change by more than 10^{-7} within 20 iterations).

Upon termination, the solution vector π was normalized so that its elements summed to unity. A normalized solution was judged feasible if convergence was reached, and if $\pi_j^{(i)} \in [0, 1], j = 1, \dots, m$ (within tolerance).

Results

Arrival Process I. For the case where the arrival rates did not vary widely with the number of customers in the queue, Tables 5 and 6 contain the total elapsed system time to termination in seconds, by method, for Systems 1 to 23 (the queues with k -Erlang service). Iteration counts are not reported, since they do not reflect the amortized cost of preconditioning where it is applied. Except for System 16, where unpreconditioned GMRES(75) stalled at iteration 1366, all methods converged to feasible solutions within 2,000 iterations.

Graphs of the time to termination against the order of Q are given in Figures 6 and 7; these show how the various methods compared in terms of the time required to solve each for each system's stationary probabilities. Figures 8 and 9 break solution time into factorization time and iteration time, respectively, for the ILU(0)-preconditioned systems; taken together, these two graphs show the extent to which factorization time dominated the total solution time. Actual NSPCG memory usage for the various methods is illustrated in Figure 10.

Similar tables and graphs for Systems 24 to 46 (queues with generalized k -Erlang service) are in Appendix B. The data for those systems are not substantially different from the results presented above. As with Systems 1 to 23, convergence to a feasible solution was attained in every case using every method, except that

Table 5. Elapsed Time to Solution (in seconds), No Preconditioner.

Number	GMRES(100)	GMRES(75)	GMRES(50)	CGS	LSQR
1	4.03	3.52	3.85	1.03	3.08
2	21.81	13.59	7.71	1.67	8.50
3	5.07	3.85	3.33	0.58	3.49
4	10.06	7.74	5.50	2.14	9.74
5	15.20	13.44	12.33	4.24	21.67
6	8.94	8.42	7.37	1.16	3.96
7	23.43	13.92	9.78	4.53	20.33
8	47.88	35.55	30.58	7.85	48.84
9	9.84	11.05	10.49	2.59	7.51
10	1.28	1.29	1.17	0.43	1.30
11	17.78	15.99	11.47	3.90	12.00
12	83.14	48.12	51.52	10.44	36.31
13	99.23	52.35	51.99	22.36	75.25
14	50.70	40.52	27.16	10.31	52.03
15	177.87	122.42	134.01	32.96	148.93
16	643.88	*	285.25	64.15	281.58
17	58.51	51.17	40.14	19.80	93.65
18	14.51	18.42	12.21	2.43	7.05
19	23.33	17.51	19.32	7.94	19.54
20	48.17	41.07	32.99	14.48	36.57
21	6.89	5.60	5.64	3.35	10.76
22	51.45	62.69	62.52	24.76	84.82
23	229.41	181.36	159.17	67.87	215.87

* stalled at iteration 1366.

Table 6. Elapsed Time to Solution (in seconds), ILU(0) Preconditioner.

Number	GMRES(100)	GMRES(75)	GMRES(50)	CGS	LSQR
1	2.43	2.40	2.31	1.52	2.37
2	3.53	3.54	5.15	2.52	4.31
3	1.65	1.67	1.66	1.61	2.13
4	5.29	5.35	5.25	3.94	6.44
5	11.87	11.61	12.80	8.27	12.64
6	3.68	3.65	3.56	2.96	4.01
7	11.72	11.54	11.89	10.30	14.13
8	26.07	27.12	25.60	20.48	32.82
9	5.57	5.65	4.77	3.07	4.77
10	1.26	1.38	1.28	1.27	1.61
11	11.78	12.85	11.95	10.93	13.88
12	37.73	38.57	37.19	33.20	42.68
13	74.04	75.34	88.08	70.03	87.00
14	62.38	67.53	65.38	63.12	70.93
15	259.72	269.11	259.00	246.23	278.11
16	561.73	553.66	560.58	519.04	596.03
17	210.80	210.72	203.12	197.53	227.15
18	5.38	4.91	4.55	3.14	4.87
19	10.38	11.44	10.39	7.82	12.47
20	21.68	24.88	20.30	14.25	25.17
21	19.54	19.98	18.80	18.35	23.71
22	166.39	163.77	160.60	160.01	173.65
23	451.15	477.29	479.14	463.97	492.06

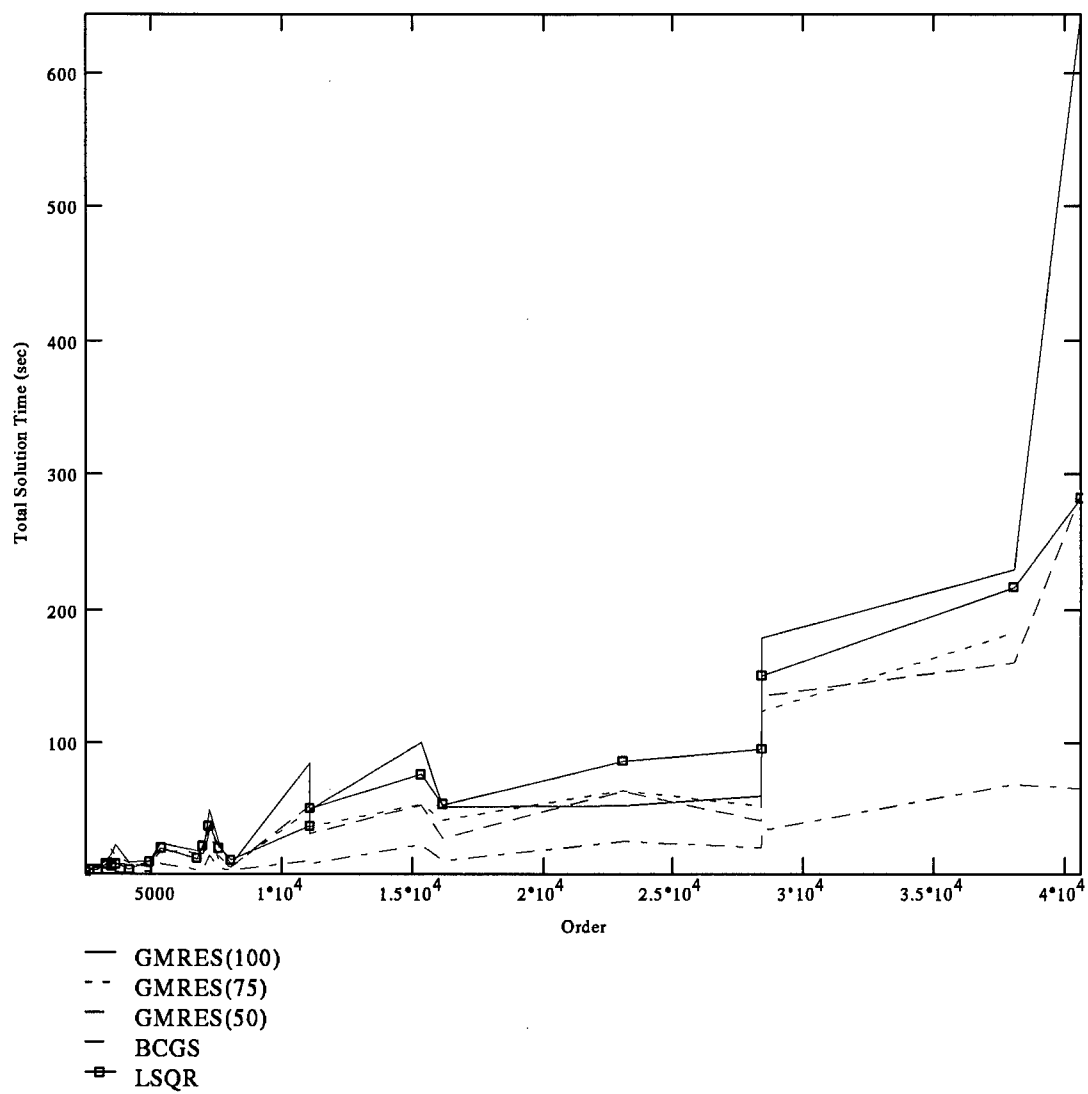


Figure 6. k -Erlang Systems: Solution Time vs. Order, No Preconditioning.

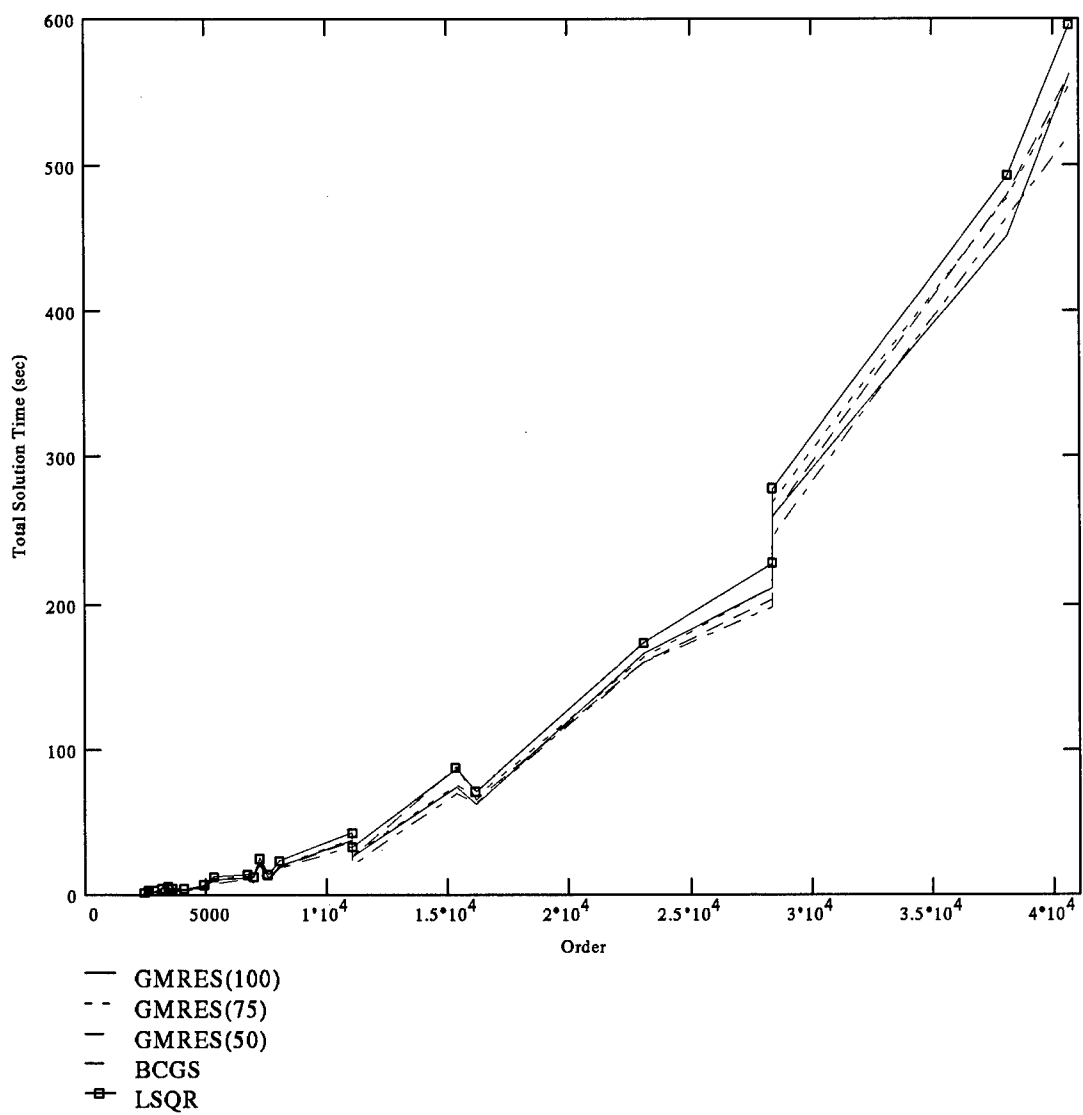


Figure 7. k -Erlang Systems: Solution Time vs. Order, ILU(0) Preconditioner.

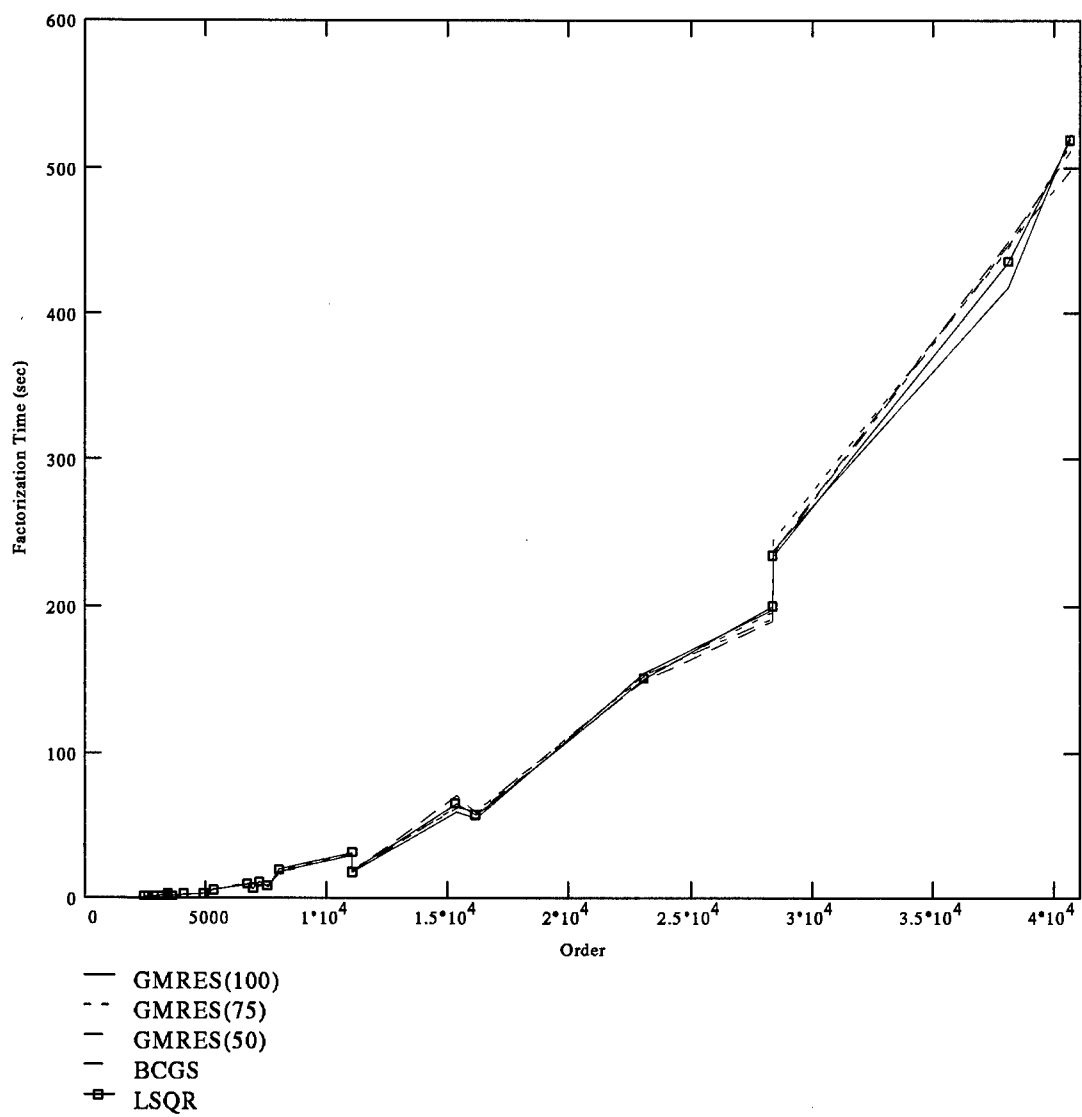


Figure 8. k -Erlang Systems: Factorization Time vs. Order, ILU(0) Preconditioner.

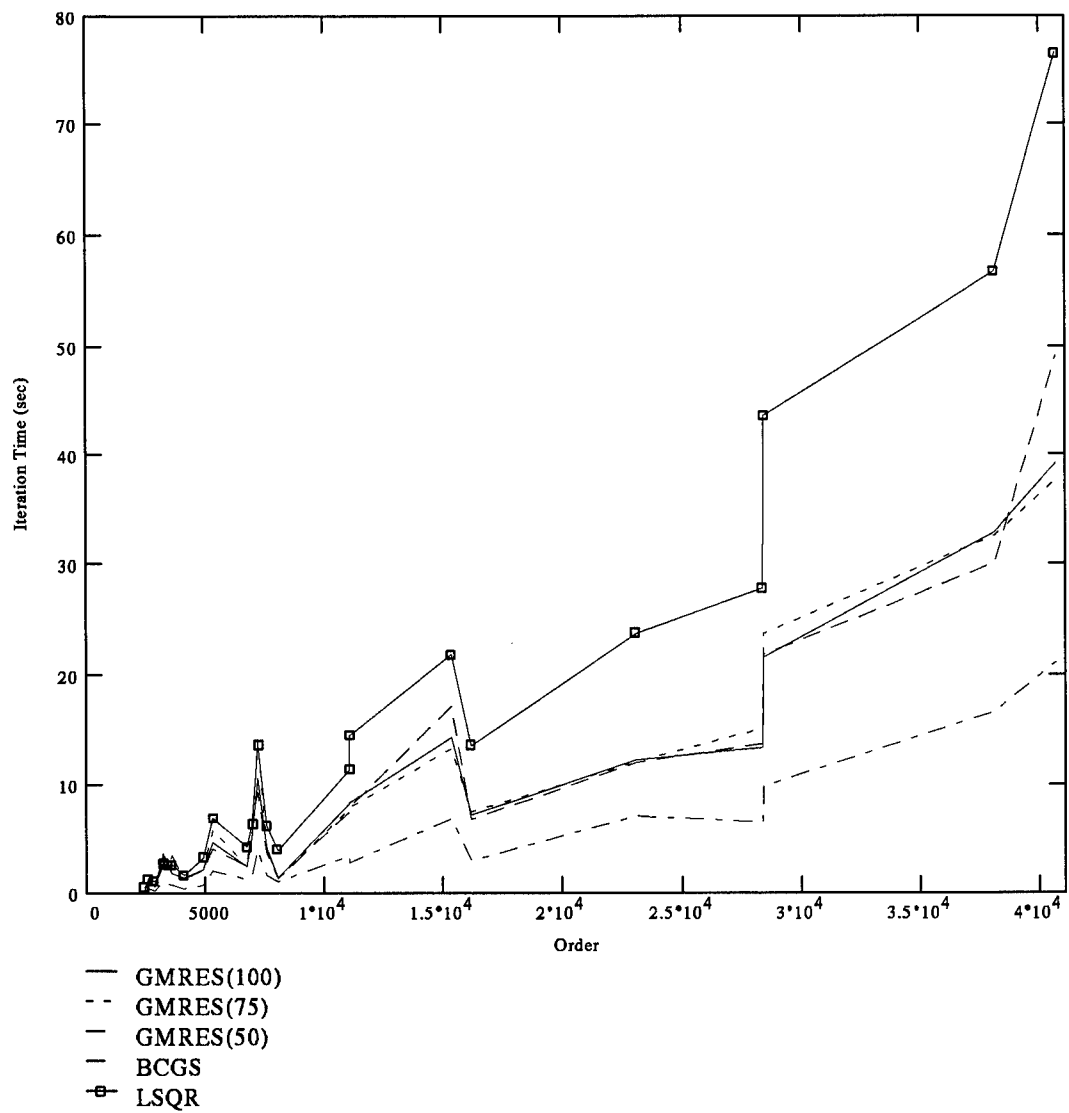


Figure 9. *k*-Erlang Systems: Iteration Time vs. Order, ILU(0) Preconditioner.

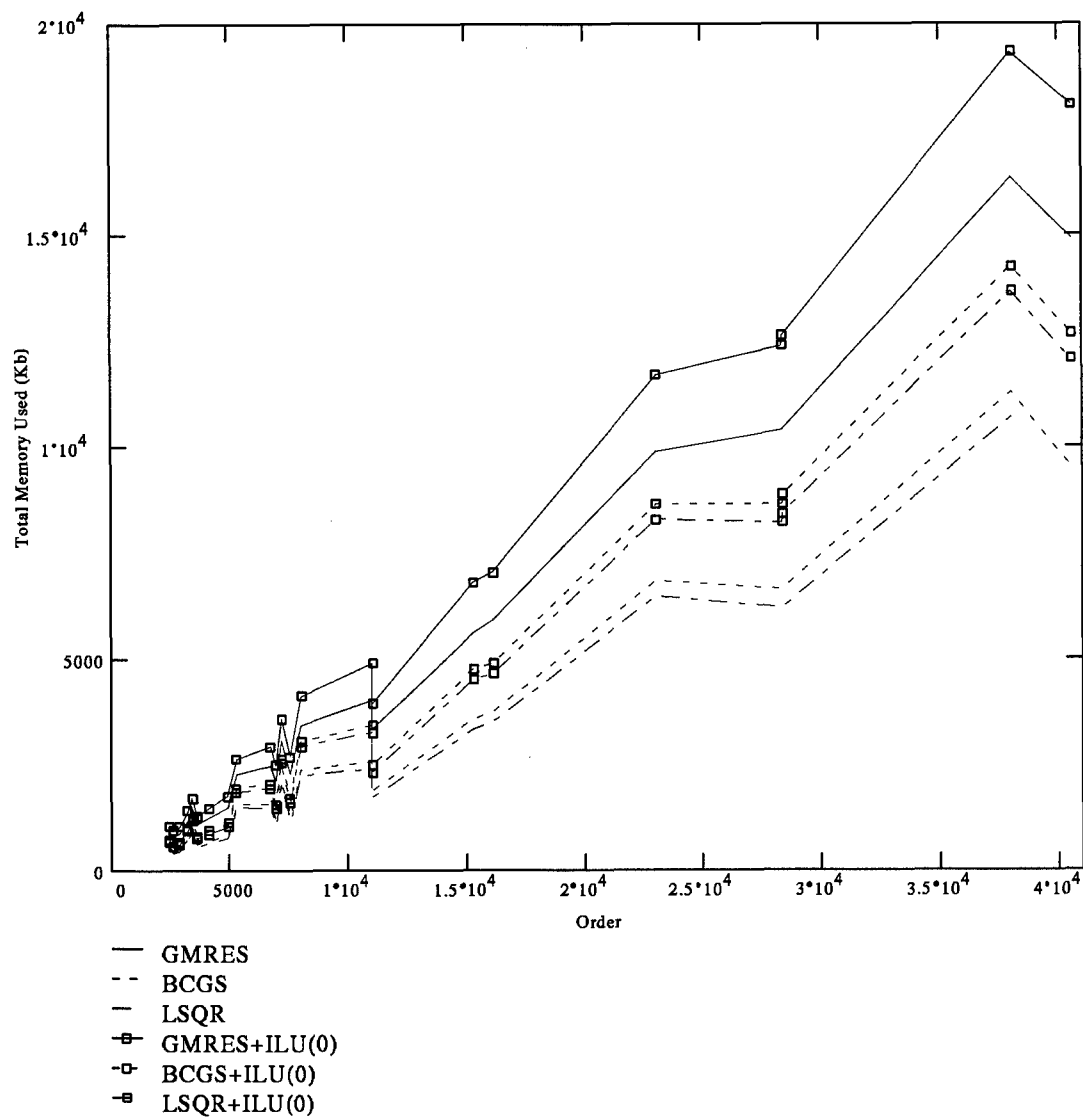


Figure 10. Memory Usage vs. Order.

unpreconditioned LSQR failed to converge within 2,000 iterations for Systems 35, 36, 38, 39, 43, 45, and 46. However, the residuals in these cases were less than 2×10^{-2} at termination and were decreasing; therefore, LSQR would likely have converged had the iteration limit been increased.

No data are reported for equation systems preconditioned using either the modified ILU(0) or the ILUTH factorizations. For all 46 systems, none of the methods tested converged when modified ILU(0) was applied. The ILUTH factorizations took about the same amount of elapsed time as their ILU(0) counterparts, but they did not significantly reduce the number of iterations to convergence significantly beyond the gain realized by using ILU(0).

There was no appreciable difference in performance for any method when $m \leq 10,000$. For $m > 10,000$, the preconditioned methods took similar amounts of time to converge; however, they tended to be outperformed by the unpreconditioned methods because of the cost of preconditioning a large Q^T matrix. Differences in performance were noticeable for the unpreconditioned methods as m increased past 10,000, with CGS dominating all methods in convergence speed. As expected, CGS also had the slowest growth in convergence speed, and required less memory than the GMRES(j) methods. CGS's stable convergence behavior was *not* expected, given the algorithm's documented tendency to diverge. This behavior is likely related to the fact that for the class of CTMCs examined in the study, π is fairly well-conditioned.

Preconditioning the system of equations using standard ILU factorization significantly sped theoretical convergence (as measured by the number of iterations required) for all of the iterative methods considered, at the expense of significantly increased memory overhead. Unfortunately, this reduction in the number of iterations did not translate into a significant solution time reduction. In fact, the cost in time of factoring a large Q^T matrix completely dominated any gains in convergence speed when $m > 10,000$.

Arrival Process II. Detailed results for the systems with the widely varying arrival rates are omitted. In the case of methods incorporating standard ILU preconditioning, convergence was stable, and the solution times and iteration counts were closely similar to those of the non-extremal cases. However, convergence problems were noted for the unpreconditioned case. For GMRES(j) and LSQR, the solutions for some systems either stalled or did not converge; unpreconditioned CGS diverged for 18 of the systems. See Table 7 for a summary of the irregular convergence behavior.

For the unpreconditioned systems that did converge, the solution algorithms performed about as well as they did for queues with the less-variable arrival rate vectors. The failure of a solution method to converge for a particular system could not be uniformly predicted by the structure of the queue being analyzed, although two general characteristics of the unstable behavior were noted:

1. For unpreconditioned methods, more k -Erlang systems converged than did their generalized counterparts.
2. Convergence behavior stabilized when the systems were preconditioned.

Further experimentation suggested that the erratic convergence behavior is insensitive to perturbations to $\pi^{(0)}$, independent of changes to the dimension of the Krylov subspace, and unaffected by scaling the entries of Q^T . Therefore, the deterioration in the convergence behavior is in all likelihood driven by ill conditioning introduced by the pathological nature of the arrival process.

Using CGS for Isolated Analysis in Marie's Method

We turn now to a demonstration of the effectiveness of using unpreconditioned CGS as part of the isolated analysis of multiserver stations. Using Marie's method, approximate stationary probabilities were calculated for four closed queuing networks, each having the general topology pictured in Figure 11.

Table 7. Convergence Behavior, Arrival Process II, No Preconditioner.

Number	GMRES(100)	GMRES(75)	GMRES(50)	CGS	LSQR
1				D	
2				D	
3					
4					
5				D	
6					
7					
8			F	D	
9					
10					
11					
12				D	
13				D	
14					
15				D	
16				D	
17					
18					
19					
20					
21					
22					
23					
24					
25				D	
26					
27			F		
28			S	D	
29					
30			F		
31		F	S	D	
32					
33					
34					
35			S	D	
36			F	D	
37					
38		S	S	D	
39	S	F	S	D	F
40			S		
41					
42				D	
43				D	
44					
45			S	D	
46					F
Total not Converging	1	3	11	18	2

S = stalled; F = failed to converge by iteration 2000; D = diverged

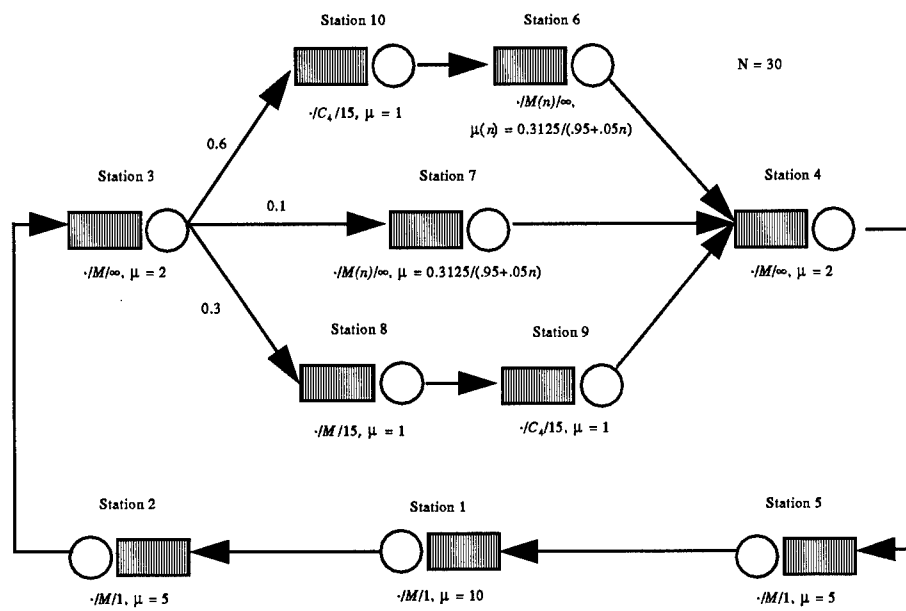


Figure 11. General Network Topology.

The networks were constructed so that both single and multiple station decompositions could be studied. In networks Ia and Ib, station 10 had a 4-Coxian service law, while its complement is made up of stations that possess product-form properties. Similarly, stations 9 and 10 in networks IIa and IIb had Coxian distributions. In networks Ia and IIa, the Coxian stations had 15 4-Erlang servers, similar to System 14 of the previous section. In Networks Ib and IIb, stations 9 and 10 each had 15 generalized 4-Erlang servers, as did System 37. The structure of the stations was chosen so that the embedded Markov chains were sufficiently large; see Table 8 for a description of stations 9 and 10 by network.

Table 8. Parameters of Stations 9 and 10.

Network	Station 9	Station 10
Ia	$\cdot/M/15$	$\cdot/E_4/15$
Ib	$\cdot/M/15$	$\cdot/GE_4/15$
IIa	$\cdot/E_4/15$	$\cdot/E_4/15$
IIb	$\cdot/GE_4/15$	$\cdot/GE_4/15$

Following the advice of Baynat and Dallery [23], the complement of the stations to be isolated was replaced by a single flow-equivalent load-dependent exponential server. The service rates for this server were determined using aggregation. The resulting equivalent networks are pictured in Figures 12 and 13.

CGS was used to solve for the stationary probabilities of the isolated station(s). The results of the experiments are reported in Table 9.

In all examples, rapid convergence was observed. The total time required to solve for the stationary probabilities of the isolated station on each iteration is consistent with the results for Systems 14 and 37 reported in Tables 5 and 18. Also, the required total solution time needed for each network is still less than the time for a single preconditioned solution for the appropriate systems. This supports the decision to omit preconditioning GCS.

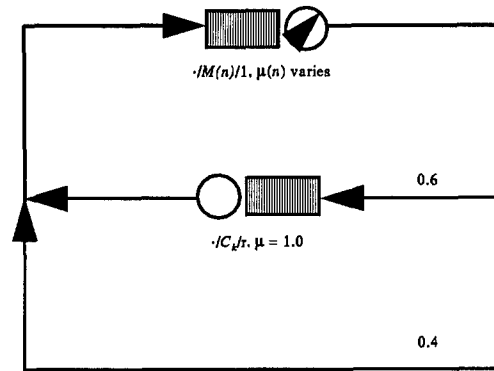


Figure 12. Equivalent Topology, Networks Ia and Ib.

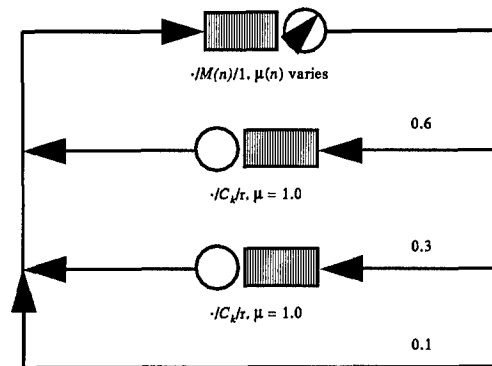


Figure 13. Equivalent Topology, Networks IIa and IIb.

Table 9. Results of Marie's Method Decomposition.

Network	Iterations	Setup Time	Solution Time	Other Tasks	Total Time
Ia	3	4.37	31.84	0.33	36.54
Ib	2	3.12	44.68	0.40	48.20
IIa	2	6.24	41.68	0.44	48.34
IIb	2	6.35	100.80	0.44	107.59

Note: All times are in seconds.

Conclusions

When the elements of the arrival rate vector of the $\lambda(n)/C_k/r/N$ queue are reasonably bounded, unpreconditioned CGS is clearly the preferred method for solving for π when the order of Q is greater than 10,000. For problem sizes between 2,500 and 10,000 states, CGS is still the fastest method, although its advantage over the others is less pronounced; still, it is to be preferred to GMRES(j) because it uses significantly less memory. Note that this disagrees with the conclusion of Philippe et al. that CGS is too unstable to be useful for solving for the stationary probabilities of CTMCs typically arising from queuing network applications [95]; this is probably because π is better conditioned for the systems considered here than for those considered in [95] (which deals more with embedded chains that are NCD).

When the elements of the arrival rate vector λ vary significantly, the choice is less clear-cut, since the performance of unpreconditioned iterative methods tends to deteriorate. The best explanation for this seems to be that the more extreme the variance in the arrival rates, the worse the conditioning of the problem. This ill-conditioning seems to be aggravated by variance within μ (the vector of service rates for the Coxian stages). Since CGS is known to be especially susceptible to convergence instability, its deficient performance in this case is not surprising; however, the experiment showed that the pathological behavior can be eliminated by applying standard ILU preconditioning before solving with CGS. The deteriorating

convergence should not be a vital concern, however, since in practical queuing network applications one is unlikely to encounter an input rate vector that varies to the extent captured in Equation 15. Still, if CGS breaks down and preconditioning is too expensive, GMRES(j) (with $j \geq 100$) can be used as a slower but more stable alternative to CGS, provided memory capacity is not a problem.

For this type of CTMC, the results show preconditioning is costly when Q^T is large; therefore, it is obviously impractical to factor Q^T during each iteration of Marie's method. It may be possible, however, to amortize the cost of preconditioning by only factoring Q^T on the first iteration of Marie's method, and then using the resulting M matrix on all subsequent calls. This method, which is sometimes used in time-dependent finite element analysis, only works well if Q^T remains relatively stable [93:23]. Unfortunately, we cannot guarantee the stability of Q^T from iteration to iteration, because the nature of the changes to the corresponding vector λ are unknown. Also, if we accept Marie's claim that his method typically converges within three to five iterations, factoring Q^T even just once would significantly increase the required solution time. It therefore seems beneficial to avoid preconditioning by factorization when solving for π for this class of CTMCs.

Modified ILU(0) decomposition is often judged superior to the standard in performance to the standard ILU(0) algorithm for certain classes of systems, particularly for finite element problems [7:41]. For this problem class, however, using modified ILU(0) significantly *worsened* convergence behavior in every case considered. When a factorization is necessary, therefore, the modified ILU(k) method should be avoided.

By using of unpreconditioned CGS to solve for the stationary probabilities of the $\lambda(n)/C_k/r/N$ queue, the class of networks that can be quickly and efficiently analyzed using Marie's method is greatly expanded. In the next two chapters, this class of networks will be expanded even further to include queuing networks containing fork and join nodes. The resulting expanded class will include the AAM.

IV. Fork-Join Queuing Networks

Introduction

Over the last decade, there has been much interest in the general class of stochastic models that incorporate synchronization constraints. These models incorporate such complicating structural factors as customer resequencing, resource sharing and concurrent processing. Research in parallel computer performance evaluation, flexible manufacturing, and telecommunications has driven the development of the theory of synchronized systems; Baccelli and Makowski give several interesting examples of applications [12], [13].

In this chapter, we consider queuing networks containing two types of synchronizing structures that were briefly introduced in Chapter I: *fork primitives* and *join primitives*. Queuing networks that contain one or both types of these structures are called *fork-join queuing networks* (FJQNs). The remainder of the chapter has three parts. First, several types of FJQNs are described. This is followed by a summary of several methods for analyzing these networks that are described in the literature. The chapter ends with a brief discussion of the published techniques and their usefulness for analyzing the airfield model presented in Chapter I.

Network Types

The Fork-Join Queue. The simplest form of a FJQN is the *fork-join queue* (also called the *split-match queue* or the *assembly-disassembly queue*). Such queues are useful for representing a system in which customers require simultaneous service at more than one station. In a fork-join model, an arriving customer enters through a fork primitive, where it splits into several identical entities. Each of these new entities, which we will call *clones* because each has the same attributes and arrival process as the original customer, immediately visits one of several independent

parallel service stations (typically, each station has a single server, but multiserver stations are possible). When each clone's service is complete, it waits in a buffer at the join primitive until all its matching clones (or *siblings*) are served; at that point, the clones are "reassembled" into the original customer, which immediately leaves the system. Figure 14 shows the directed graph of a fork-join queue.

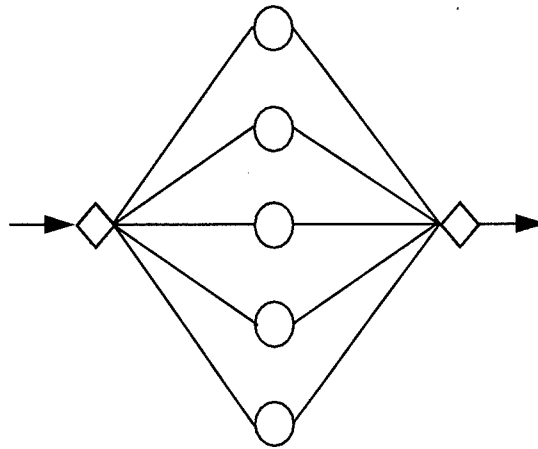


Figure 14. Sample Fork-Join Queue Topology.

The manner in which arriving customers split into clones is called the *loading pattern*. This loading pattern may be deterministic, in which case every customer sends a clone to every server, or the number and destination of the clones may vary according to some probability law.

If the servers in a fork-join queue have identically-distributed service times, they are said to be *homogeneous*, and the queue is labeled *symmetric*. If the service time distributions differ, the queue is termed *asymmetric* and is considered to have *heterogeneous* stations. In the literature, the servers are usually assumed to have infinite waiting capacity. The service disciplines are often first-come-first-served (FCFS), but the processor-sharing discipline has also been considered.

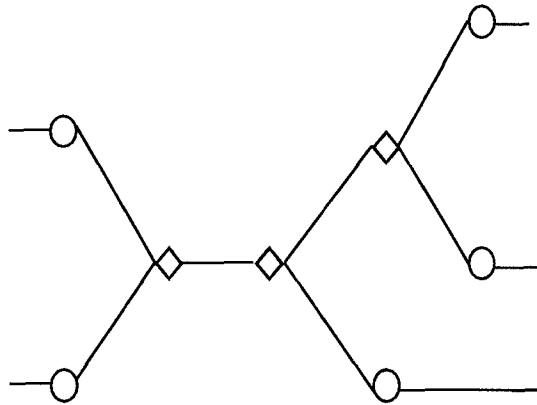


Figure 16. Sample AFJQN Topology.

Review of the Literature

Motivation. This section presents a comprehensive view of FJQN research that has been published since the 1960s. The purpose for including this section is two-fold. First, the section is intended to enlighten the reader on these developments, and to define the state of the art in the topic area. Second, it is shown that no published work considers FJQNs of the same class as the airfield model network, and that an analysis technique for this model must be proposed and developed.

Simple Fork-Join Queues.

Exact results. To date, exact performance measures have been published only for a fork-join queue with two single-server stations. In an early paper on the subject, Mandelbaum and Avi-Itzhak [77] derive exact performance measures for the case where either the arrival stream or the service distributions are deterministic.

However, many methodologies published since Reference [77] assume Poisson arrivals, exponential service, or both.

Rao and Posner [98] derive the stationary joint probability distribution of the queue lengths in a fork-join queue with a Poisson arrival stream, two heterogeneous, exponentially-distributed servers and a deterministic load pattern. They impose a limit on the capacity of one of the servers, and then solve the resulting system numerically; the solution for the unbounded system is found by allowing the limit to be arbitrarily large. In contrast to this numerical approach, Flatto and Hahn [49] derive the analytical closed form of the joint stationary probability distribution. Flatto [48] uses these findings to study the mutual dependence of the two queue lengths and to derive the stationary form of their conditional distributions. Brun and Fayolle [28] build on the work of Flatto and Hahn by deriving the probability distribution of the response time and its Laplace-Stieltjes transform. Ding [45] uses a linear-algebraic approach to derive the probability distribution of the interdeparture time.

Some authors have considered systems that are more general in character than the two-server Markovian fork-join queue. Baccelli [8] extends the results of Flatto and Hahn [49] to allow for homogeneous servers with general service laws; the stationary joint queue length distribution for the heterogeneous, general-service case is derived by DeKlein [41]. Zhang [132] considers a two-server Markovian fork-join queue in discrete time, and constructs the generating function for the joint stationary queue length distribution and the Laplace-Stieltjes transform of the joint waiting time distribution; he also derives the response time distribution for the case where the servers are homogeneous.

Bounds on the response time. Because the state space becomes unmanageably large when there are more than two parallel servers in a fork-join queue, exact solutions quickly become intractable. This fact has led several researchers

to seek alternative means of characterizing queue performance. One such approach is the construction of computable bounds on the expected response time. Using the theory of stochastic order relations [11], [102:251–283], [119:1–37], Baccelli and Makowski [10], [12], [13] derive such bounds for a fork-join queue with a general arrival process, general (possibly heterogeneous) service distributions and a deterministic loading pattern. If T_n is the response time of the n th customer to visit the fork-join queue, the Baccelli-Makowski upper bound on $E[T_n]$ is

$$E[T_n] \leq E[\max_{1 \leq i \leq k} \hat{T}_n^i] \quad (16)$$

where \hat{T}_n^i is the response time of the n th customer in an independent, single-server queue with the same arrival process as the fork-join queue and the service time distribution of the i th server in the fork-join queue. A lower bound on $E[T_n]$ is

$$E[\max_{1 \leq i \leq k} \tilde{T}_n^i] \leq E[T_n] \quad (17)$$

where \tilde{T}_n^i is the response time of the n th customer in an independent, single-server queue with one of the following structures:

1. The new queue has the same arrival process as the fork-join queue, but it has a deterministic service rate equal to the mean service rate of the i th server in the fork-join queue.
2. The new queue has the same service time distribution as the i th server in the fork join queue, but it has a deterministic arrival rate equal to the mean arrival rate to the fork-join queue.

Baccelli and Makowski show that these transient bounds also hold in steady state.

Baccelli, Makowski and Schwartz [14] extend these results to allow for a general probabilistic loading pattern. Kumar and Shorey [69] use Baccelli and Makowski's bounding techniques to develop bounds on the expected response time for a sym-

metric fork-join queue with general service and Poisson arrivals, where the loading pattern has a multinomial distribution.

Approximate performance measures. Computing bounds on the response time of a fork-join queue requires knowledge of the response time distribution for each server. Since these distributions are often difficult to obtain, several authors have proposed heuristic approximations for the queue's performance measures as an alternative to computing the bounds. One early approximation technique is that of Rao and Posner [98], who consider the asymmetric Markovian fork-join queue with two servers. They impose a virtual capacity limit on all the servers but one, and then use a numerical approach to solve for the approximate steady-state joint queue length probabilities. No evaluation of the approximation's performance is provided for queues with more than two servers.

Nelson and Tantawi [85], [86] consider a Markovian fork-join queue with arrival rate λ and k (≥ 2) servers with homogeneous service rate μ . They approximate the steady-state mean response time by appropriately scaling the response time of an independent M/M/1 queue with the same arrival and service rates. The scaling factor, which depends on k , λ and μ , is derived using a mixture of theoretical and empirical techniques. For the cases considered, this approximation method yields a relative error of less than 5 percent compared to simulation results.

Duda and Czachórski [46] approximate the performance of a symmetric Markovian fork-join queue with two servers by a single-server system with a load-dependent service rate. For a given load, this rate is found by calculating the throughput of a closed network containing only the fork-join queue and a customer population equal to the desired load. The approximate expectations of queue length and sojourn time are shown to be $3/2$ times as great as the equivalent measures for an M/M/1 queue; the approximate expected response time is also shown to be equal to the upper bound derived by Nelson and Tantawi [85], [86] for a queue with two servers.

For the same type of fork-join queue, Nelson, Towsley and Tantawi [88] approximate the expected steady state response time by that of a $M^X/M/c$ queue. They use the results to study a variety of task splitting schemes in parallel processing systems. The quality of the approximation is not investigated.

Kim and Agrawala [61] consider a fork-join queue with two homogeneous servers. They derive an approximate expression for the response time by conditioning on the virtual waiting time of the n th arrival, then evaluating that expression for sufficiently large n . Explicit results for various utilization rates are presented for three cases:

1. Exponential interarrival times, exponential service.
2. Exponential interarrival times, 2-Erlang service.
3. Hyperexponential interarrival times, exponential service.

Kim and Agrawala show that the approximate expected response times lie between the Baccelli-Makowski bounds, but they present no error analysis.

Balsamo and Donatiello [17] develop an approximation of the steady-state mean response time that can be used for both symmetric and asymmetric Markovian systems. They propose two heuristics for reducing the model's state space to a manageable dimension. For the resulting approximate system, they use matrix-geometric solution techniques to get the stationary joint queue length distribution. The authors compare exact and approximate results for a two-server, symmetric queue over various utilization rates. They report a maximum relative error in the approximation of less than 2 percent (for a utilization rate of 0.9).

Several authors explore the use of diffusion theory to approximate performance measures. Knessl [65] formulates and solves the diffusion equations for the system considered by Flatto and Hahn, and derives approximations for the stationary queue length distributions and the residual busy period. Varma and Makowski [128] pro-

pose a class of heuristic approximations to the limiting expected response time for a symmetric fork-join queue with general arrival and service times; comparison of these techniques to simulation results for a selection of sample systems yields a relative error no greater than 14 percent (in most of the examples considered, the error is under 10 percent). Varma [127] extends these methods to permit analysis of a fork-join queue with embedded resequencing nodes, with which he models the response time of a time-stamp ordering algorithm; he reports relative errors of 5 percent or less.

Thomasian and Tantawi [120] approximate the limiting expected response time in a symmetric fork-join queue with general service times by modeling the expected synchronization delay as a low-order polynomial in ρ (the utilization rate at a station). The form of the polynomial is determined by first simulating the system of interest at various values of ρ , then estimating the polynomial coefficients using standard interpolation techniques. While the performance of the response time approximation is not studied, the authors report a maximum relative error in the approximate synchronization delay of less than 10 percent; in most of the cases, the relative error is 3 percent or less. The quality of the approximation improves as the number of servers increases.

Rommel [101] derives the exact steady-state expected response time for a $M^X/G/1$ queue with processor sharing service discipline. He suggests this as a model for a parallel processing system where the jobs share processors. In other words, Rommel's result can be used to calculate the response time in a symmetric fork-join queue with a deterministic load pattern, where each server is a central processor shared by the clones that visit it.

Other Performance Studies. Towsley, Rommel and Stankovic [123], [124] consider fork-join queues with Poisson arrivals and both exponential and generalized exponential service distributions. In their study, they compare the performance of

these queues under two types of service discipline: first-come-first-served (FCFS) and processor sharing. According to the results they present, FCFS yields shorter response times and queue lengths unless the service distribution has high variability.

Rao [97] studies the departure process of a two-server asymmetric Markovian fork-join queue. He shows that the interdeparture times have a phase-type distribution in steady state, then uses this fact to calculate the coefficient of variation for a variety of server utilizations. The author observes that the coefficient is always very close to unity. From this, he deduces that the steady-state departure process is approximately Poisson.

Avi-Itzhak and Halfin [4] consider a fork-join queue with k servers and k customer classes, where customers in class i visit i out of k servers. They show that, if a Class 1 customer (one that doesn't clone) has the smallest mean service time, the limiting mean customer response time does not increase if that customer is given simple nonpreemptive priority at its server over clones of other classes. For some simple cases, the authors establish conditions under which giving nonpreemptive priority to a customer class causes the limiting mean response time to increase. They label their results preliminary.

Nelson and Towsley [87] investigate the impact of the imposition of various priority schemes on the performance of an asymmetric Markovian fork-join queue. They find that customer class response times are uniformly smaller under a task preemption policy (where arriving customer clones displace any lower priority clones already in service).

Several authors consider a symmetric fork-join queue where customers are split into a random number of clones that are then scheduled for service. These clones are matched to some subset of the available servers based on the size of the waiting lines at each station, so that response time is minimized. According to Makowski and Nelson [76], if the utilization rate is low, the optimal scheduling policy is to spread the clones as evenly as possible among the servers; however, for moderate to

high utilizations, the clones should be assigned to the server with the shortest queue length. Setia, Squillante, and Tripathi [109] extend Makowski and Nelson's work by performing a quantitative assessment of a range of scheduling policies. They validate Makowski and Nelson's finding that the workloads of the servers should be balanced when utilization is low; however, they recommend that customer clones should be scheduled at a progressively smaller number of servers as the utilization rate rises.

Gün and Makowski [56] examine the same system described in the previous paragraph, except that they also require customers leaving the queue to be restored to the sequence in which they arrived. A Markov decision process (MDP), where the cost per stage is the total customer response time (including the resequencing delay), is used to find the optimal schedule of clones to servers. The authors find that the optimal scheduling policy is to balance the workloads at the servers as quickly as possible.

Open Networks With Fork and Join Primitives. Baccelli, Massey and Towsley [12], [13], [15], [16] formulate bounds on the expected response time in an AFJQN by generalizing the results in [10]. As with the simple fork-join queue, both bounding systems are constructed by neglecting the stochastic coupling induced by the synchronization constraints; the lower bounding system also imposes deterministic interarrival or service distributions. For PFJQNs, the bounding method involves treating the subnetworks along each fork independently, just as the individual servers in a fork-join queue. Baccelli and Massey develop complex recursion relations to calculate the bounds for more general network topologies. They also establish stability conditions. These results are extended to cover networks with more general priorities and customer precedence relations by Baccelli and Liu [9] and Liu and Baccelli [75].

Gershwin [53], [54] proposes a decomposition method for AFJQNs with finite buffers and unreliable servers. His approach centers on analyzing each finite buffer in isolation. For two sample systems, each with eight servers, Gershwin's algorithm

produces approximate performance measures that fall within 95 percent confidence intervals derived by simulating the systems. The author observes that numerical instability appears in the algorithm as the buffer capacities approach infinity [54].

King [63] shows that, for a PFJQN where each subnetwork is composed of an arbitrary number of serial exponential servers, the exact distribution of customer response time is a mixture of Erlang distributions. Because of the rapid increase in the cardinality of the state space, King shows that it is impractical to estimate the parameters of the exact distribution. To overcome this problem, he develops a heuristic parameter estimation method that is based on the Central Limit Theorem. For the hypothetical systems King considers, the heuristic provides accurate parameter estimates when compared to Monte Carlo simulation.

Duda and Czachórski [46] generalize their approximation technique (see above) for a symmetric fork-join queue to accommodate networks of fork-join queues with arbitrary numbers of phase-type servers. Their method, which they implement in the specialized queuing analysis computer language QNAP2, is a numerical algorithm that replaces each fork-join construct by a single-server queue with a load-dependent service rate. The rates are calculated by aggregating servers in a manner similar to that described in the previous section. No error analysis is reported.

Konstantopoulos and Walrand [66] derive the stability conditions for an AFJQN with an arrival process that is a general point process. They also generalize these stability results to allow for random routing in the network.

Ammar and Gershwin [2] consider FJQNs with blocking; they prove a theorem that establishes structural and probabilistic equivalence conditions for this class of networks when the service times are all exponentially distributed. Dallery and his coauthors [40] generalize these results to arbitrary FJQNs with blocking. In addition, they establish conditions for symmetry and reversibility, and prove the concavity of the throughput function for certain classes of input processes.

Nguyen [91],[92] considers a feedforward network of fork-join queues with deterministic routing. She uses a reflected Brownian motion (RBM) technique to derive a heavy-traffic diffusion approximation for expected network response time. She also provides a numerical algorithm for calculating the stationary probability density of the approximating RBM process.

Campos, Chiola, and Silva [30] propose using free-choice Petri nets [84] to model single-class networks with synchronization constraints. By using a combination of the theories of Petri nets and classical queuing networks, they derive bounds on the throughput levels, mean queue lengths, and mean response time. They also provide polynomial-time algorithms for calculating these bounds. No investigation of the quality of the bounds is reported.

Rajaraman and Morgan [96] study the same class of FJQNs as King, except that they allow each server to have a general service time distribution. They assume a generalized exponential distribution for the response time along each subnetwork, then use the Baccelli-Massey technique to compute an upper bound on network response time. Compared to simulated response times for a variety of candidate systems, the approximation technique yields relative errors of 10 percent or less.

Closed Networks Containing Fork-Join Subnetworks. Several studies have been published considering closed systems that contain fork-join queues or networks. Among the earliest of these is the paper by Almeida and Dowdy [1], who model a sequential/parallel processor as a closed network containing a single exponential server (to model the sequential execution process) in series with a symmetric fork-join queue with two or more exponential servers (representing the parallel execution process). They construct an approximating closed system by replacing the fork-join queue by two single-server queues in tandem: a typical fork-join processor, followed by a delay station that mimics the synchronization delay. Mean value analysis (MVA) [100] is used to derive performance measures and system throughput for the approx-

imating network. The throughput approximation compares favorably with that of the simulated throughput of the original system (less than 1 percent relative error for various sample systems).

Liu and Perros [73] consider a closed network containing an exponential single-server queue in series with a single fork-join queue with an arbitrary number of exponential servers (this system is similar to the one studied by Almeida and Dowdy). They create an approximating system by aggregating all but one of the parallel servers into a composite server with a load-dependent service rate. The exact steady-state joint probability distribution of the queue lengths for the approximate system is obtained by constructing and solving its global balance equations; this distribution is used as an approximate limiting distribution for the original system. Since the relative error of throughput is observed to increase linearly as the number of servers increases, Liu and Perros incorporate an empirical scaling factor that stabilizes the relative error of the throughput approximation at less than 3 percent. In a related paper [74], Liu and Perros formulate an alternate approach to obtaining an exact probability distribution for this system when the fork-join queue has three servers; their approach uses a combination of decomposability theory [36] and Gauss-Seidel iteration.

Rao and Suri [99] develop a heuristic approximation technique based on MVA to study a closed, multiclass queuing network containing a fork-join queue with parallel exponential servers. Their method is based on two key assumptions:

1. When a customer clone arrives at a server in the fork-join queue, it finds the system in a state that would be seen at a random point in time with a customer (i.e., itself) removed from the system.
2. The response times at the servers in the fork-join queue are mutually independent exponential random variables.

The authors report relative errors in the approximate throughput (compared to simulation results) of less than 8 percent; the maximum relative error in the mean queue lengths is 16 percent.

Jenkins [58] extends Rao and Suri's results to allow for multiserver exponential stations and a probabilistic loading pattern. The multiserver stations are analyzed using the load-dependent versions of the MVA recursion equations [35:182–183]. Jenkins deals with probabilistic loading by first conditioning on whether or not a representative customer visits the embedded fork-join structure, then calculating the cycle time for each possible customer path. Dietz and Jenkins [44] improve the algorithm's performance by conditioning on the ordered subset of servers a customer visits. The resulting algorithm is a very accurate heuristic: for the example considered, the relative error of the approximations with respect to simulation results is less than one percent for the throughput levels and three percent for the mean queue lengths. Hackman [57] extends the Dietz and Jenkins method to accommodate general service laws by using heavy-traffic approximations; the results appear promising for networks with moderate to high utilization levels.

Baynat and Dallery [21], [22] propose a strategy for analyzing single-class, closed networks with embedded fork-join subnetworks, deterministic load patterns and multiserver stations with general service time distributions. The authors approximate the system of interest by a network with a product-form solution [19]; the performance measures of the second system are used as approximations for those of the original network. The product-form network is constructed as follows:

1. Replace all FCFS service stations with nonexponential service times by an approximately equivalent single server with load-dependent, exponential service times. The service rates can be calculated using either aggregation (as in Algorithm 8.1 of Lazowska et al. [72:161]) or Marie's method [78], [79].

2. Replace each fork-join subnetwork by an approximately equivalent single server with load-dependent, exponential service times. To get the service rates, analyze the fork-join subnetwork as an isolated, closed network with a separate customer class for each clone. The throughput levels of the isolated system are used as the load-dependent service rates of the replacement server.

For the two sample systems studied, this decomposition approximation yields maximum relative errors of 2 percent for throughput and 9 percent for response time when compared to simulation results, with most relative errors less than 2 percent.

In subsequent papers [24], [25], Baynat and Dallery extend their method for use with multiclass networks where the fork and join primitives are the only structures visited by more than one customer class. The core of this approach is to analyze the join primitive in isolation using Marie's Method. The accuracy of this approximation is similar to that of its single-class counterpart. A similar approach is used by Di Mascolo, Frein, Baynat and Dallery to analyze a multi-stage production line system [42], [43].

Discussion of the Literature.

Recall from Chapter 1 that the system to be investigated in this research project is a capacitated FJQN containing a nested fork-join construct multiserver FCFS queues. Currently, the state of the art in FJQN analysis does not provide a method for bounding or approximating the performance measures of this type of system. Existing methods are inappropriate for analyzing the system of interest because they cannot handle all of the following attributes:

1. One or more stations have multiple servers operating under an FCFS service discipline.
2. The loading patterns are probabilistic.
3. Fork-join constructs may be nested within one another.

Clearly, a method is needed for accurate, efficient analysis of networks having these three features. The work of Baynat and Dallery provides the best starting point for the development of such a method, for the following reasons:

1. It assumes a capacitated system.
2. It uses a straightforward, product-form approach.
3. It allows, at least in theory, the incorporation of multiserver stations.

Product-Form Approximation of Closed Fork-Join Queuing Networks.

Introduction. Since product-form approximations of FJQNs are foundational to the discussion in the next chapter, they are examined in detail in this section. The material that follows is an abstract of Baynat and Dallery's work, and represents the state of the art in product-form approximations for fork-join queuing networks. The full theoretical development is contained in References [21] and [22].

Suppose a queuing network containing a fork-join construct has been feasibly partitioned using the guidelines presented in Chapter II. Then one of the subsystems will necessarily contain the fork-join construct. This subsystem can be analyzed using either aggregation or Marie's method to get approximate conditional throughput levels. Figure 17 depicts a representative closed fork-join network, which will be used to aid the exposition in this section.

Analysis Using Aggregation. If the conditional throughput levels are to be obtained using the aggregation technique, the isolated fork-join subnetwork would be formed by short-circuiting its complement, as in Figure 18. Baynat and Dallery propose transforming this isolated network by treating each clone as a separate customer class, and combining the fork node and join buffer into a multi-class synchronization station with a deterministic zero service time and synchronized departures. The equivalent network is shown in Figure 19. To analyze this network, the multiple-

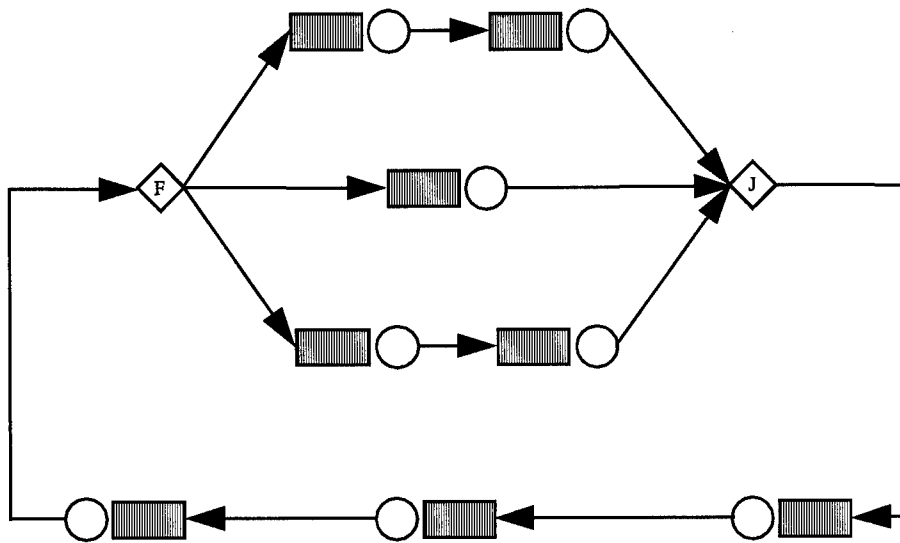


Figure 17. Representative Fork-Join Queuing Network.

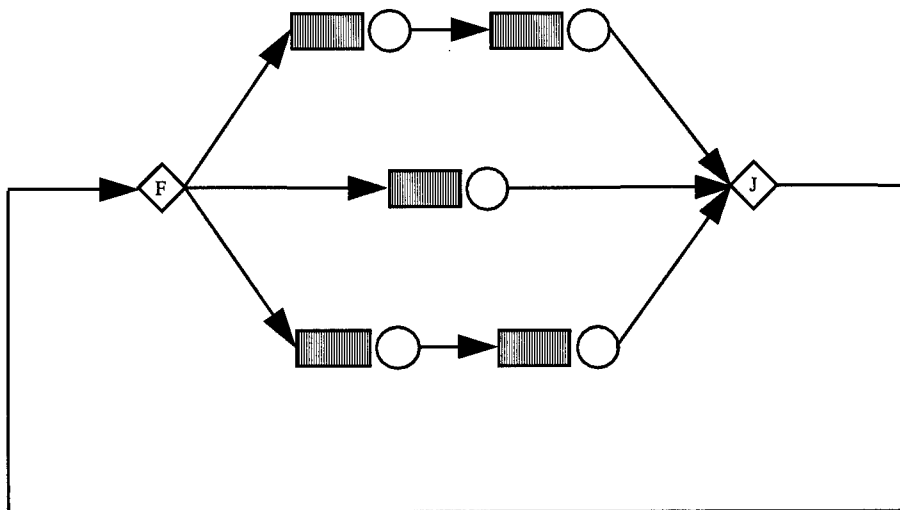


Figure 18. Isolated Fork-Join Subnetwork, Aggregation Method.

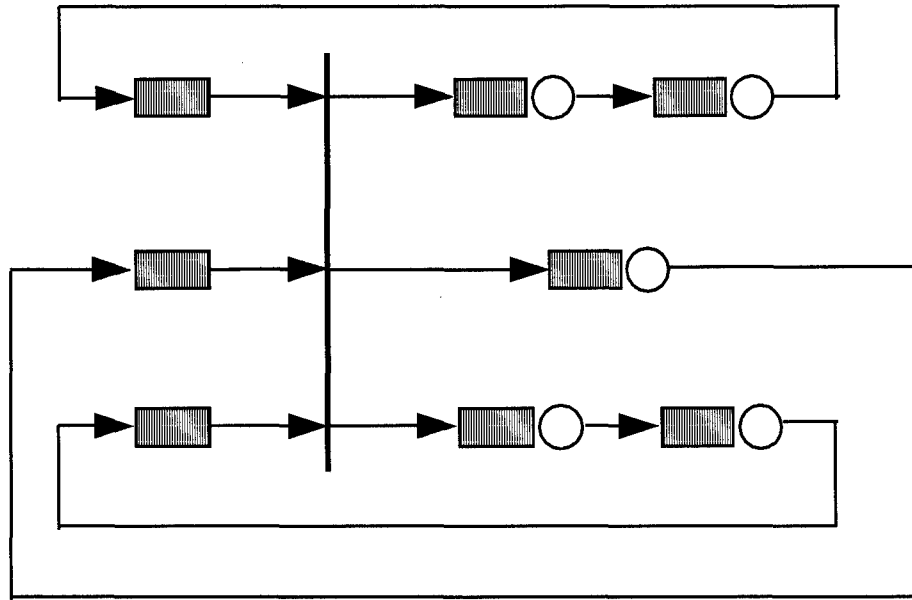


Figure 19. Transformed Subnetwork, Aggregation Method.

chain extension to Marie's Method is used, with the synchronization station (and perhaps other stations) being analyzed in isolation. Baynat and Dallery give closed form steady-state probabilities for a two-class synchronization station. If there are more than two classes, the analysis in isolation can be carried out by formulating and solving the embedded continuous-time Markov chain. In the case where there are many more than two forks in the construct, the analysis may be simplified by aggregating classes, as alluded to previously.

Analysis Using Marie's Method. If Marie's Method is to be used, the isolated fork-join subnetwork would be formed as an open, capacitated network with load-dependent Poisson arrivals; this would, in turn, be reformulated as the equivalent closed network shown in Figure 20. Notice that the station representing the Poisson arrival process has mean service rate $\mu_0(n) = \lambda(N - n), n = 1, \dots, N$. Baynat and Dallery's transformation of this network is similar to the aggregation case, except

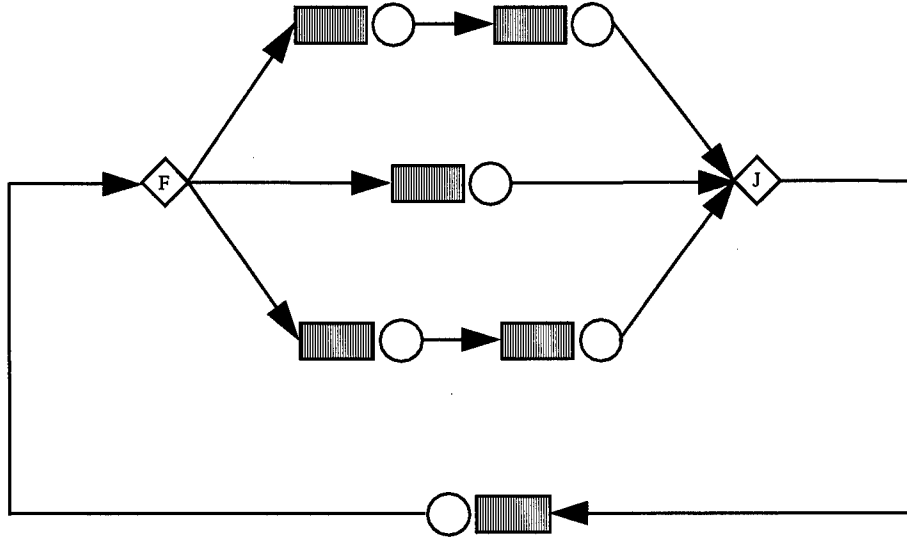


Figure 20. Isolated Fork-Join Subnetwork, Marie's Method.

that the join buffer is combined with the external Poisson arrival process to form a timed synchronization station. This station has mean service rate $\mu_0(n_0)$, where $n_0 = \min_r n_{0r}$ and n_{0r} is the number of clones of class r waiting in the join buffer. The equivalent network is shown in Figure 21; as in the case of aggregation, Marie's method is used to analyze this network. The synchronization station is analyzed in a manner similar to that described for aggregation.

Necessary Theoretical Extensions

To be able to analyze the airfield model, two extensions to existing theory need to be made. First, the above procedure for decomposing FJQNs must be modified to allow for probabilistic load patterns in the fork-join constructs. Also, a strategy must be outlined for dealing with nested fork-join constructs. The next chapter describes proposed solutions to these problems.

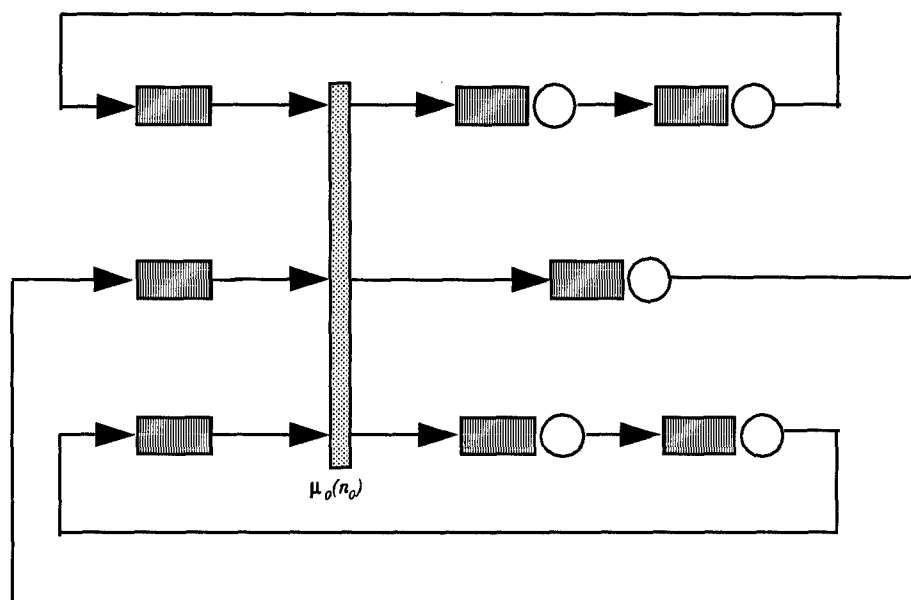


Figure 21. Transformed Subnetwork, Marie's Method.

V. A Product-Form Approximation Technique for Closed Nested Fork-Join Queuing Networks With Probabilistic Load Patterns

Introduction

As we saw in Chapter IV, the state of the art in product-form approximations of closed fork-join queuing networks is the multiple-chain reformulation method of Baynat and Dallery [21], [22]. Recall that Baynat and Dallery assume deterministic forking; that is, a customer always visits every subnetwork connected to a fork node. Also, nested fork-join constructs (the case where any fork-join subnetwork (FJSN) contains another FJSN) are not explicitly treated except in Reference [24], where one is given as an example of a system for which product-form approximations fail. The problem of interest, however, belongs to a class of networks with nested FJSNs, where each FJSN has a probabilistic load pattern. In this chapter, an approximation strategy is developed that extends the results of Baynat and Dallery to accomodate this larger class of networks.

The “Short-Circuit” Approximation

Description. In Chapter IV, we saw that Baynat and Dallery analyze an isolated FJSN by reformulating it as a closed, multiple-chain network with a synchronization station; this station has varying forms depending on the strategy used to decompose the original network (aggregation or Marie’s method). Marie’s method for multiple-chain networks (MMMC), which is introduced in Chapter II, is used to analyze the reformulated subnetwork in isolation.

Now suppose the FJSN of interest has a probabilistic load pattern. If this is true, then it is obvious that a customer may completely bypass one or more of the embedded subnetworks with positive probability. An intuitive way to model this behavior is to introduce feedback loops into the appropriate chains in the isolated,

reformulated subnetwork; these feedback loops allow a customer to bypass all stations in the chain and return immediately to the synchronization station. This strategy, which we will call the “*short-circuit*” (*SC*) *approximation*, is graphically illustrated in Figure 22 (note that Marie’s method has been used to formulate the isolated subnetwork).

The SC approximation requires an additional assumption not imposed by Baynat and Dallery: the customer clones in the isolated FJSN can match interchangeably. SC produces approximate results because the matching assumption may not be true for the original network model. The rationale for the approach is that the resulting expected increase in throughput induced by the assumption of interchangeability should partially offset the effect of the independence assumptions required by MMMC.

Analyzing the Synchronization Station. The feedback loops described above can be dealt with in one of two ways: they can be incorporated into the embedded Markov chain of the synchronization station (*internal feedback*), or they can be left as part of the product-form approximation to the isolated FJSN (*external feedback*). In the latter case, all that is required is to adjust the visit ratios for the isolated FJSN. When the feedback loops are incorporated into the embedded chain, however, the chain has a slightly different formulation from that described in References [21] and [22]. The following discussion develops the structure of the embedded chain for the case of two forks. Extension to the case of three or more forks is straightforward; to reduce complexity, these Markov chain formulations can be used with the class aggregation scheme proposed by Baynat and Dallery [24], [25].

Aggregation. We first consider the case where aggregation is used to decompose the original network. Let n_i = the number of class i customers in the join buffer ($i = 1, 2$), and let (n_1, n_2) be the state of the synchronization station. Clearly, the only feasible states in the embedded chain are those for which $n_1 = 0$

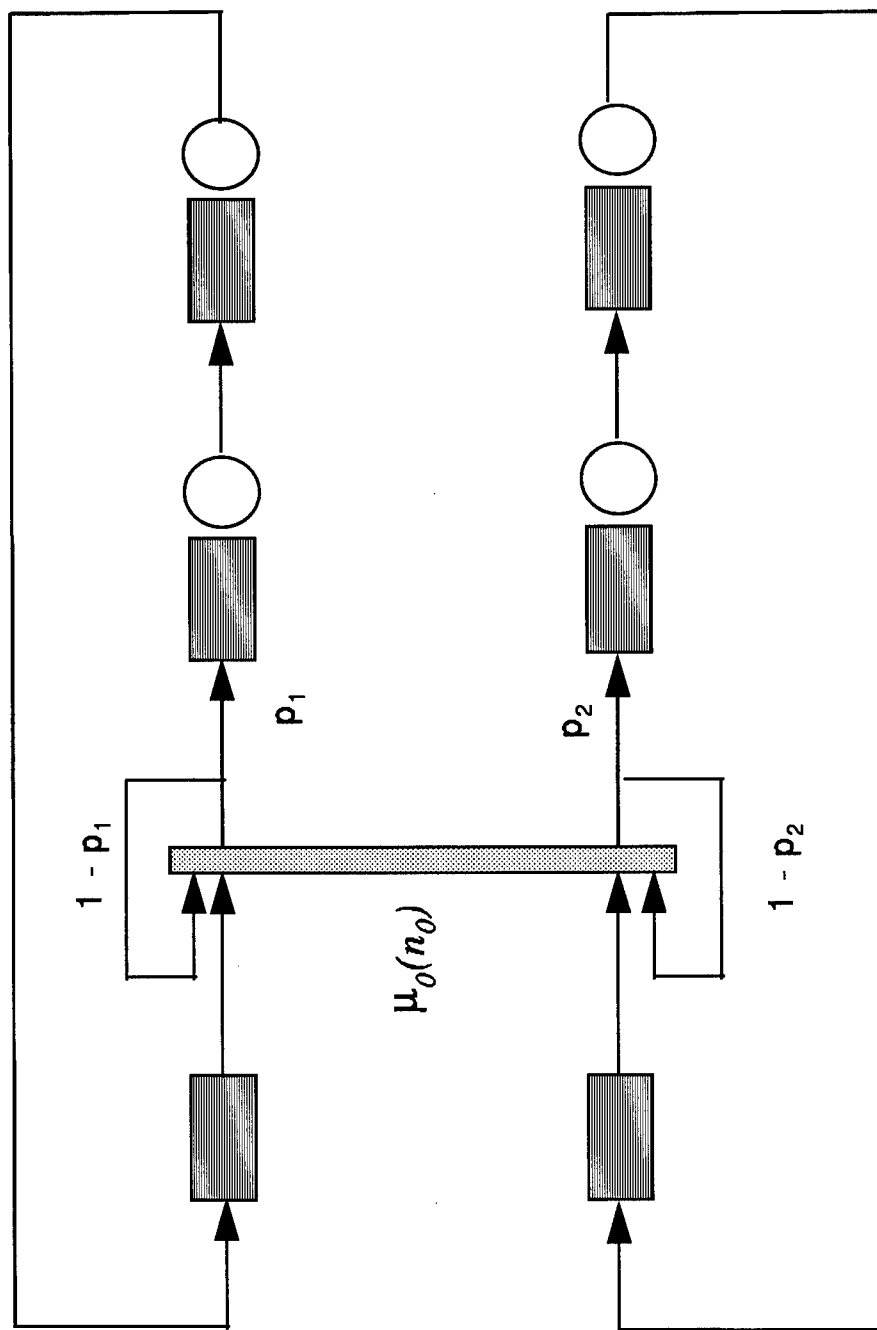


Figure 22. Using SC Approximation With Marie's Method.

or $n_2 = 0$ (or both). State transition behavior is complicated by the fact that one or both matching customers can return to the join buffer in zero time following a match.

Assume, for the sake of discussion, that $n_1 = 0$ and $0 \leq n_2 \leq N$ (where N is the network population). Then the system state changes in one of the following ways:

1. If $0 \leq n_2 < N$, a class 2 customer arrives.
2. If $n_2 = 0$, a class 1 customer arrives.
3. If $0 < n_2 \leq N$, a class 1 customer arrives, and causes between 0 and n_2 class 2 departures prior to departing in synchronization with the final class 2 departure.
4. If $0 \leq n_2 \leq N$, a class 1 customer arrives, causes n_2 departures, and remains at the synchronization station.

Define $\lambda_i(n_i)$ as the arrival rate of class i customers, and p_i as the probability that a class i customer leaves the synchronization station. Further, let

$$P_1(j) = \Pr[\text{a class 1 customer causes } j \text{ class 2 departures before leaving}], \quad j < n_2$$

and let

$$P_1^0 = \Pr[\text{a class 1 customer causes } n_2 \text{ class 2 departures, and stays at station}]$$

The states with which $(0, n_2)$ communicates, as well as the appropriate transition rates, are in Table 10.

Table 10. State Transitions, Aggregation.

Transitions to	Rate	Conditions
$(0, n_2 - j)$	$P_1(j)\lambda_1(0)$	$n_2 \in [0, N], j \in [0, n_2]$
$(0, 1)$	$P_1^0\lambda_1(0)$	$n_2 \in [1, N]$
	$\lambda_1(0)$	$n_2 = 0$
$(0, n_2 + 1)$	$\lambda_2(0)$	$n_2 \in [0, N - 1]$

We need to derive the probabilities $P_1(j)$ and P_1^0 . To get $P_1(j)$, we condition on the number of feedback loops required to produce j class 2 departures:

$$P_1(j) = \sum_{i=j}^{\infty} \Pr[j \text{ class 2 departures} \mid \text{class 1 departure after } i\text{th loop}] \cdots \\ \cdots \times \Pr[\text{class 1 departure after } i\text{th loop}]$$

Clearly

$$\Pr[\text{class 1 departure after } i\text{th loop}] = (1 - p_1)^{i-1} p_1$$

and

$$\Pr[j \text{ class 2 departures} \mid \text{class 1 departure after } i\text{th loop}] = \binom{i-1}{j-1} (1 - p_2)^{i-j} p_2^j$$

Therefore,

$$P_1(j) = \sum_{i=j}^{\infty} \binom{i-1}{j-1} (1 - p_2)^{i-j} p_2^j (1 - p_1)^{i-1} p_1 \\ = \sum_{i=1}^{\infty} \binom{j-1+i-1}{j-1} (1 - p_2)^{i-1} p_2^j (1 - p_1)^{j-1+i-1} p_1$$

$$\begin{aligned}
&= \frac{p_2^j (1-p_1)^{j-2} p_1}{(1-p_2)[1-(1-p_1)(1-p_2)]^{j-1}} \times \\
&\quad \sum_{i=1}^{\infty} \binom{j-1+i-1}{j-1} [(1-p_1)(1-p_2)]^i [1-(1-p_1)(1-p_2)]^{j-1} \\
&= \frac{p_2^j (1-p_1)^{j-2} p_1}{(1-p_2)[1-(1-p_1)(1-p_2)]^{j-1}} \{1 - [1-(1-p_1)(1-p_2)]^{j-1}\} \quad (18)
\end{aligned}$$

since each term in the infinite series is a negative binomial density [47:110].

Since the probability that a class 1 customer remains in the system after n_2 class 2 departures is $(1-p_1)^i$, we have that

$$\begin{aligned}
P_1^0 &= P_1(n_2) \frac{1-p_1}{p_1} \\
&= \frac{p_2^{n_2} (1-p_1)^{n_2-1}}{(1-p_2)[1-(1-p_1)(1-p_2)]^{n_2-1}} \{1 - [1-(1-p_1)(1-p_2)]^{n_2-1}\} \quad (19)
\end{aligned}$$

The probabilities $P_2(j)$ and P_2^0 are easily derived by exchanging subscripts in Equations 18 and 19.

To efficiently derive the transition rate matrix Q for the embedded Markov chain, we first order the $2N+1$ states as follows: $(N, 0), (N-1, 0), \dots, (1, 0), (0, 0), (0, 1), \dots, (0, N-1), (0, N)$. For state (row) s , we can then assign nonzero values to the appropriate columns of Q as outlined in Table 11.

Marie's Method. When Marie's method is used, the embedded chain with feedback loops is somewhat simpler to formulate because of the nonzero delay after each synchronization. In this case, states exist where both n_1 and n_2 are nonzero. The states with which (n_1, n_2) communicates, together with the appropriate transition rates, are in Table 12.

Table 11. Column Entries For Row s of Q , Aggregation.

Column Index	Rate	Conditions
$s + j$	$P_2(j)\lambda_2(0)$	$s \in [1, N],$ $j = 1, \dots, N + 1 - s$
$N + 2$	$P_2^0\lambda_2(0)$	
$s - 1$	$\lambda_2(N + 1 - s)$	$s \in [2, N + 1]$
$s - j$	$P_1(j)\lambda_1(0)$	$s \in [N + 2, 2N + 1],$ $j = 1, \dots, s - N - 1$
N	$P_1^0\lambda_1(0)$	
$s + 1$	$\lambda_1(s - N - 1)$	$s \in [N + 1, 2N]$

Table 12. State Transitions, Marie's Method.

Transitions to	Rate	Conditions
$(n_1 + 1, n_2)$	$\lambda_1(n_1)$	$n_1 < N$
$(n_1, n_2 + 1)$	$\lambda_2(n_2)$	$n_2 < N$
$(n_1 - 1, n_2)$	$p_1(1 - p_2)\mu_0(\min[n_1, n_2])$	$n_1, n_2 > 0$
$(n_1, n_2 - 1)$	$(1 - p_1)p_2\mu_0(\min[n_1, n_2])$	
$(n_1 - 1, n_2 - 1)$	$p_1p_2\mu_0(\min[n_1, n_2])$	

The transition rate matrix Q can be efficiently generated by ordering the states first on n_1 , then on n_2 : $(0, 0), (0, 1), \dots, (N, N - 1), (N, N)$. When this ordering scheme is followed, the non-zero entries in row s of Q can be generated according to the rules in Table 13.

Table 13. Column Entries For Row s of Q , Marie's Method.

Column Index	Rate	Conditions
$s + N + 1$	$\lambda_1(n_1)$	$n_1 < N$
$s + 1$	$\lambda_2(n_2)$	$n_2 < N$
$s - N - 1$	$p_1(1 - p_2)\mu_0(\min[n_1, n_2])$	$n_1, n_2 > 0$
$s - 1$	$(1 - p_1)p_2\mu_0(\min[n_1, n_2])$	
$s - N - 2$	$p_1p_2\mu_0(\min[n_1, n_2])$	

Extension to Nested Fork-Join Queuing Networks. Although nesting of fork-join constructs is not explicitly addressed in the open literature, it makes sense to deal with them by applying Baynat and Dallery's unified theory in a hierarchical manner. All that is necessary is that the assumptions required by the unified theory be satisfied by the network partitions at all levels of the hierarchy.

For the purpose of illustration, suppose we have a FJQN with two FJSNs, one nested within the other. In this case, hierarchical decomposition requires isolated analysis of structures at the following three levels:

1. The nested fork-join construct.
2. The FJSN embedded in the nested construct, together with the outer synchronization station.
3. The inner synchronization station.

Note that other individual stations may need to be analyzed in isolation at any of these three levels. This hierarchical process is illustrated in Figure 23.

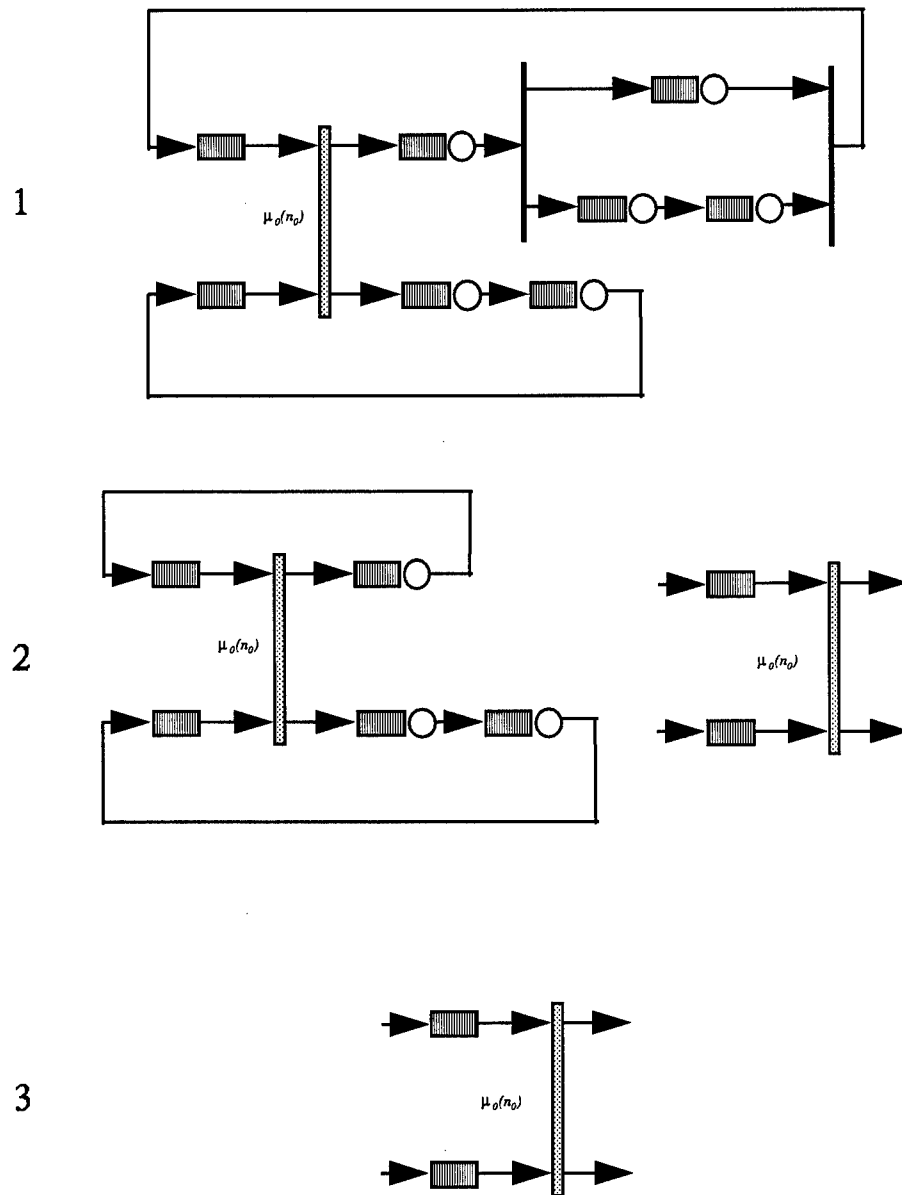


Figure 23. Hierarchical Decomposition of a Nested FJQN.

Computational Experience

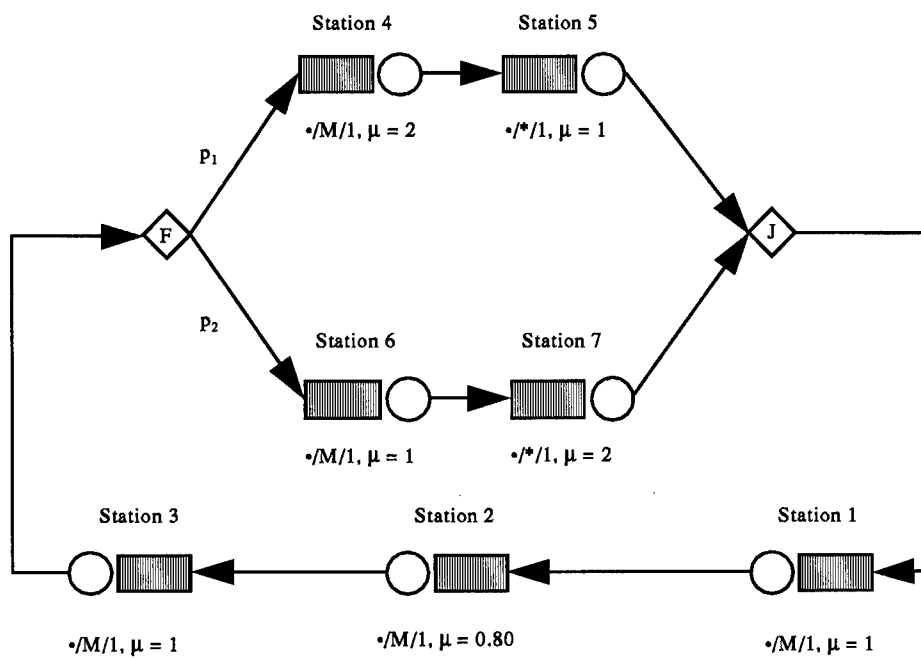
Case Study 1: FJQNs With Probabilistic Forking.

Overview. A numerical study was conducted to examine the performance of the SC approximation method when applied to FJQNs without nested fork-join constructs. SC, with both internal and external feedback, was combined with both aggregation and Marie's method to produce a set of four candidate approximation strategies:

1. SC with internal feedback, using Marie's method to decompose the original network (SCMI).
2. SC with external feedback, using Marie's method to decompose the original network (SCME).
3. SC with internal feedback, using aggregation to decompose the original network (SCAI).
4. SC with external feedback, using aggregation to decompose the original network (SCAE).

The default stopping criteria (Equations (5) and (9)) were used for Marie's method.

Eighteen different configurations of the same underlying network topology were studied. The topology, which was motivated by the airfield flow problem, is shown in Figure 24. The attributes varied between configurations were the probabilities p_1 and p_2 , the service time distributions of Stations 5 and 7, and the network population. The service rates were chosen so that the response time along the subnetworks in the fork-join construct would be balanced, thus allowing the effect of varying the load pattern and the service distributions to be more transparent. The specific configurations are enumerated in Table 14.



* Service law varies

Figure 24. Basic Network Topology, Case Study 1 Representative Systems.

Table 14. Case Study 1: Representative System Configurations.

No	N	Stn 5	Stn 7	(p_1, p_2)
1	5	2-Erlang(1.0)	2-Erlang(0.5)	(0.5,0.5)
2	10			
3	20			
4	5	2-Cox(0.25,2.5,0.3)	2-Cox(0.25,2.5,0.1)	
5	10			
6	20			
7	5	Exponential(1.0)	Exponential(1.0)	
8	10			
9	20			
10	5	2-Erlang(1.0)	2-Erlang(0.5)	(0.9,0.1)
11	10			
12	20			
13	5	2-Cox(0.25,2.5,0.3)	2-Cox(0.25,2.5,0.1)	
14	10			
15	20			
16	5	Exponential(1.0)	Exponential(1.0)	
17	10			
18	20			

Each method was applied to each representative system to compute the expected throughput at Station 1 and the expected queue lengths at all seven stations. The performance of each method was judged on the basis of the relative error between the performance measures it produced and the true values of those measures. Since exact analyses of the systems were impossible, "truth" was taken to be a simulation point estimate whose 95 percent confidence interval half-width was less than or equal to 10^{-2} , regardless of the magnitude of the point estimate¹.

Results. Based on the results of the numerical study, SCMI provided the most consistently accurate approximations to the network performance measures. SCAE was competitive with SCMI when used to estimate expected throughput and expected queue lengths at Stations 1 to 3; however, this method failed to produce accurate expected queue lengths for stations inside the fork-join construct (that is, the relative errors were all between 30 and 1200 percent, with most errors on the high end of that scale). This may be due to the fact that aggregation strips information on higher moments from the queue length distributions that cannot be accurately restored during "disaggregation."

The performance measures produced by SCME were dominated by those from SCMI; as with SCAE, SCME gave grossly inaccurate queue length estimates for Stations 4 through 7 (implying that mere adjustment of the routing probabilities in the aggregate network is not sufficient to preserve the distributional information of the queue lengths inside the fork-join construct). SCAI was also ineffective as an approximation strategy, since it in every case was dominated by SCAE, and in many cases yielded performance measures with well over ten percent relative error, especially for small populations. For these reasons, results are not presented here for either SCAI or SCME.

¹This choice was based on the reasoning that since changes to queuing network parameters of such a small magnitude typically have little practical significance, it makes little sense to insist on greater precision.

Relative errors in the approximate throughput at Station 1 are presented in Figure 25 for balanced loads (Systems 1 through 9); Figure 26 shows the relative errors for systems with unbalanced loads (Systems 10 through 18). Numerical results are listed in Appendix C. Clearly, both SCAE and SCMI approximated throughput accurately. Neither method dominated the other except for Systems 14, 15, and 16, where SCAE was clearly more accurate.

Figures 27 to 40 depict the relative error in the queue length approximations at the seven stations. For balanced loads, the performance of the two methods for stations outside the fork-join structure tended to be similar, with both methods generally increasing in accuracy as population increased. For unbalanced loads, the methods behaved more irregularly for Stations 1, 2, and 3, with SCMI tending to dominate SCAE. For the high-variance cases (Systems 13–15), performance tended to deteriorate as population increased. The methods failed for System 15 at Station 1 (SCMI) and Station 2 (SCAE).

For stations inside the fork-join structure, SCAE failed completely, producing relative errors of between 30 and 1,200 percent. In contrast, SCMI performed very well, except when Station 7 had a high-variance Coxian service law (Systems 4 through 6 and 13 through 15). In these cases, however, the absolute error was found to be a fraction of a customer, and was always less than one percent of the network population.

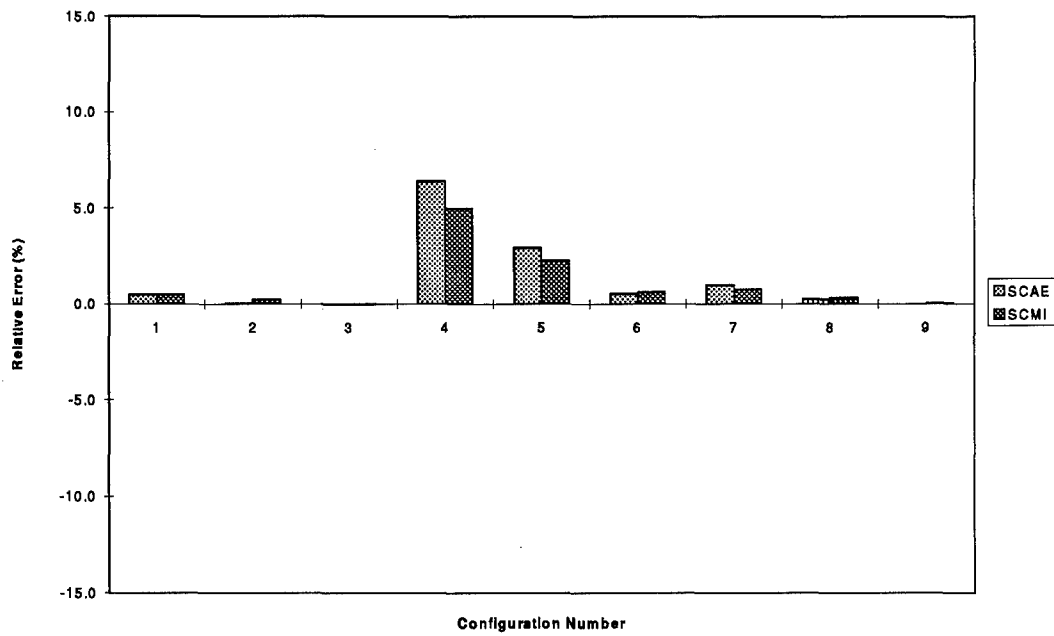


Figure 25. Case Study 1: Expected Throughput at Station 1, Systems 1–9.

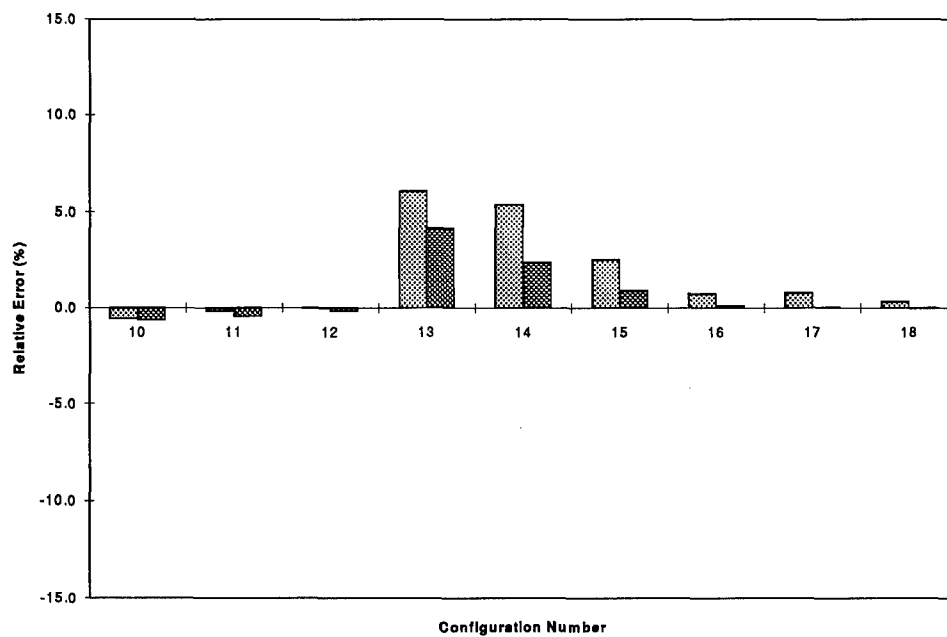


Figure 26. Case Study 1: Expected Throughput at Station 1, Systems 10–18.

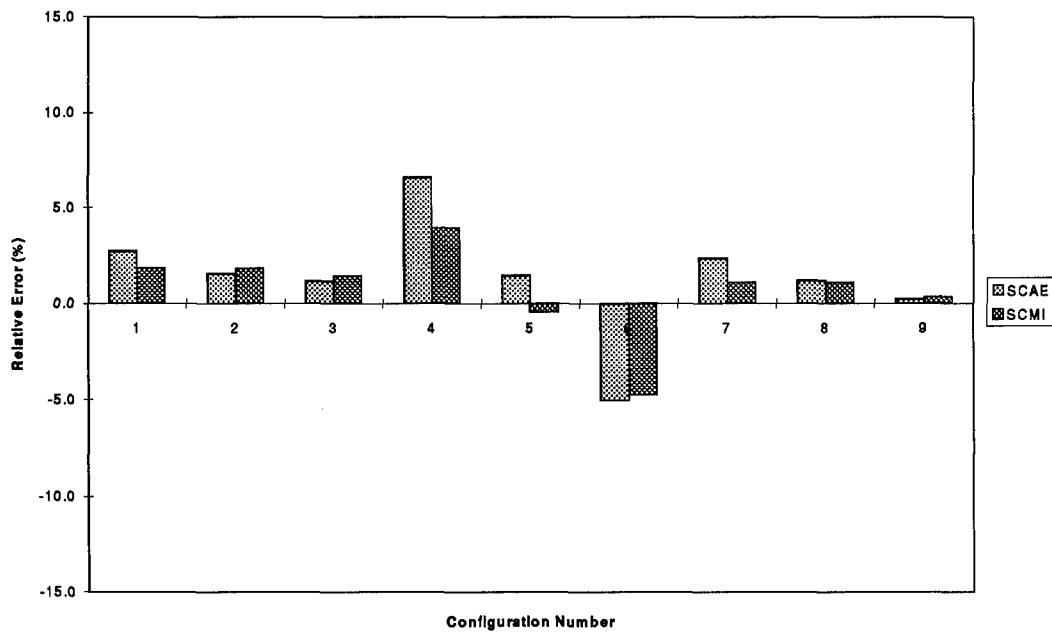


Figure 27. Case Study 1: Expected Queue Length at Station 1, Systems 1-9.

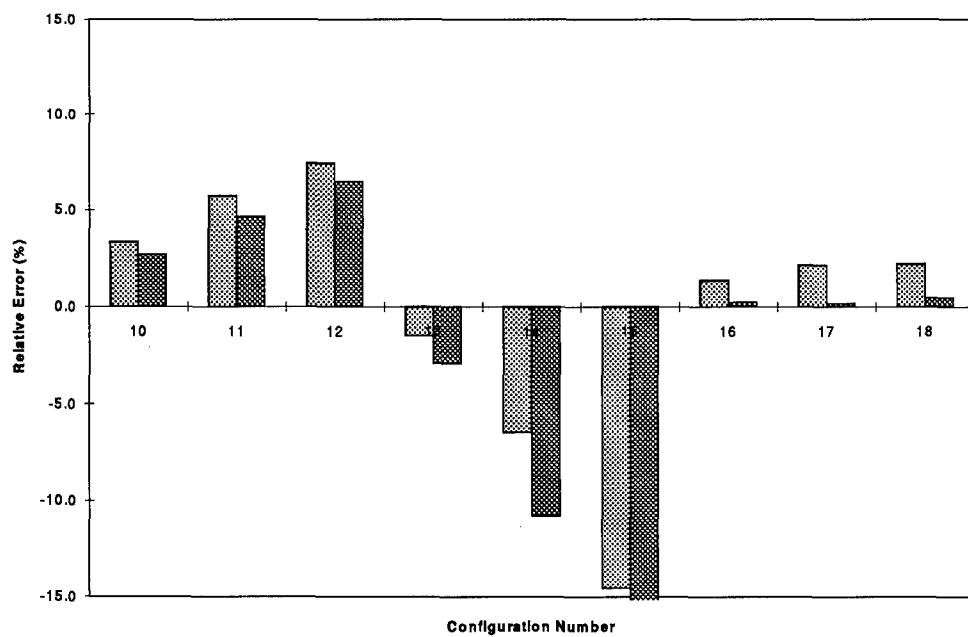


Figure 28. Case Study 1: Expected Queue Length at Station 1, Systems 10-18.

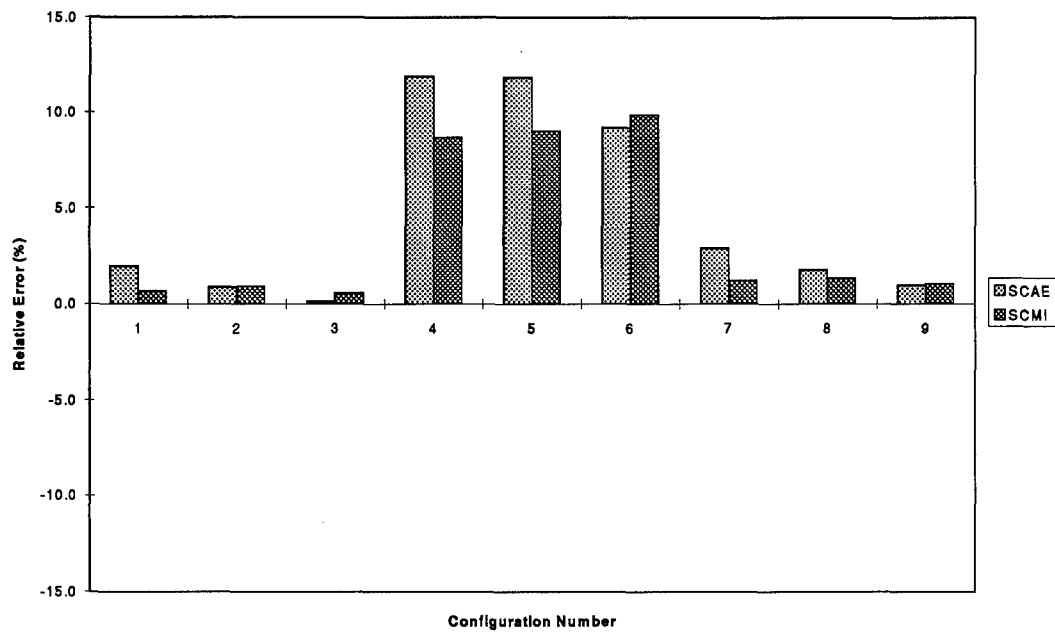


Figure 29. Case Study 1: Expected Queue Length at Station 2, Systems 1–9.

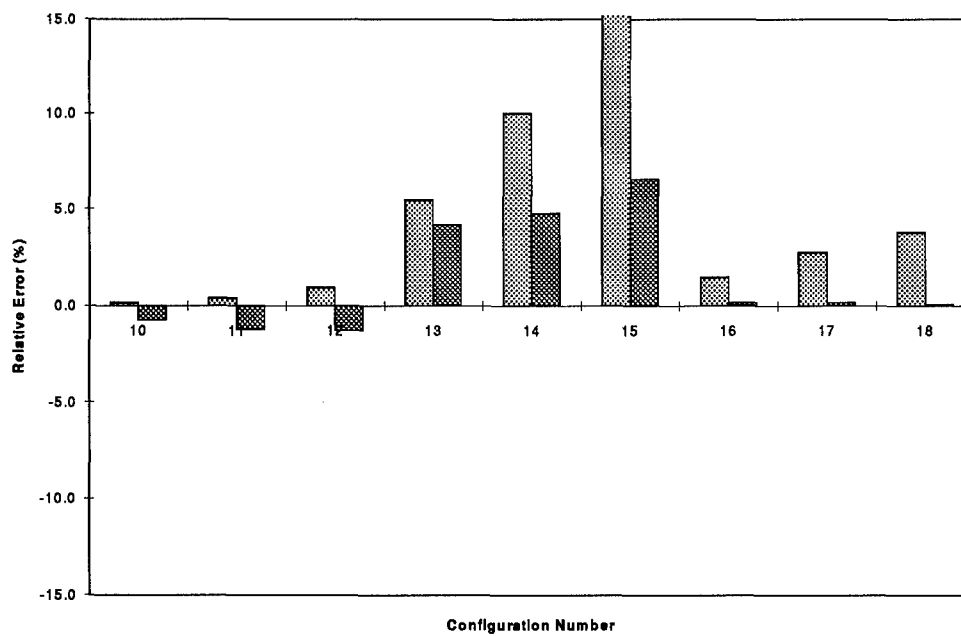


Figure 30. Case Study 1: Expected Queue Length at Station 2, Systems 10–18.

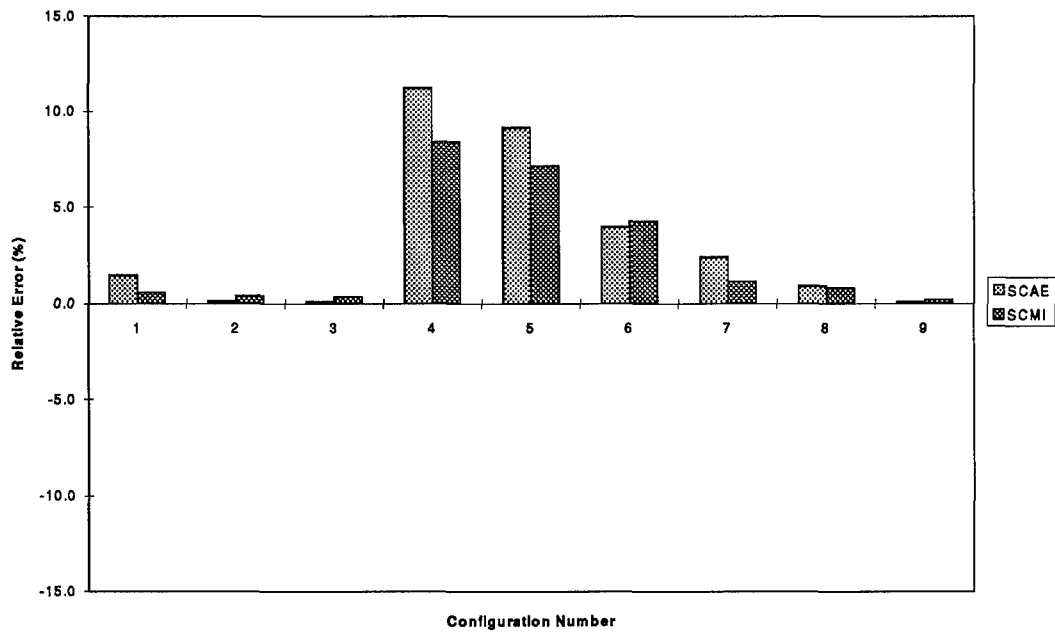


Figure 31. Case Study 1: Expected Queue Length at Station 3, Systems 1-9.

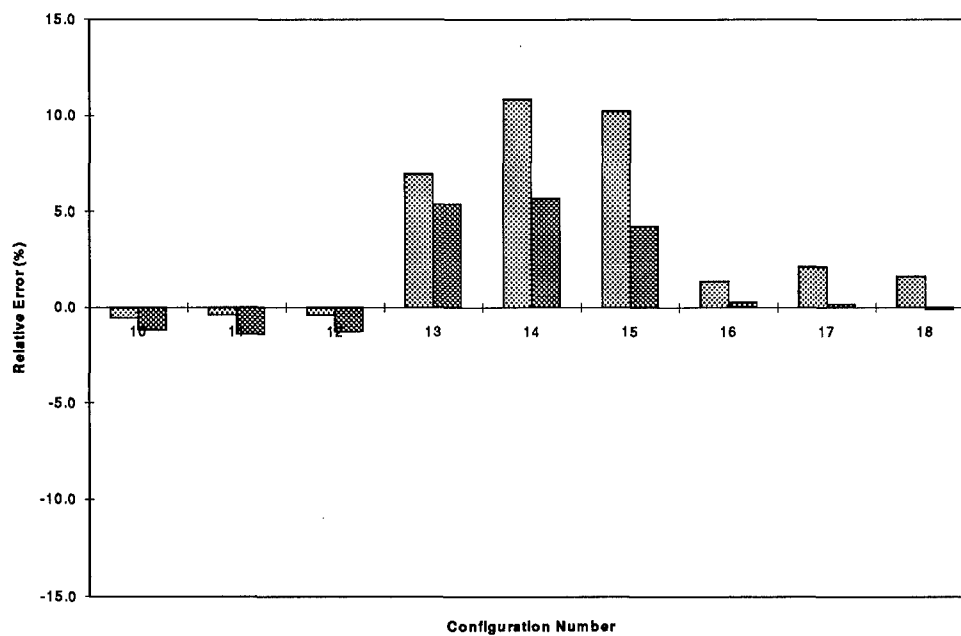


Figure 32. Case Study 1: Expected Queue Length at Station 3, Systems 10-18.

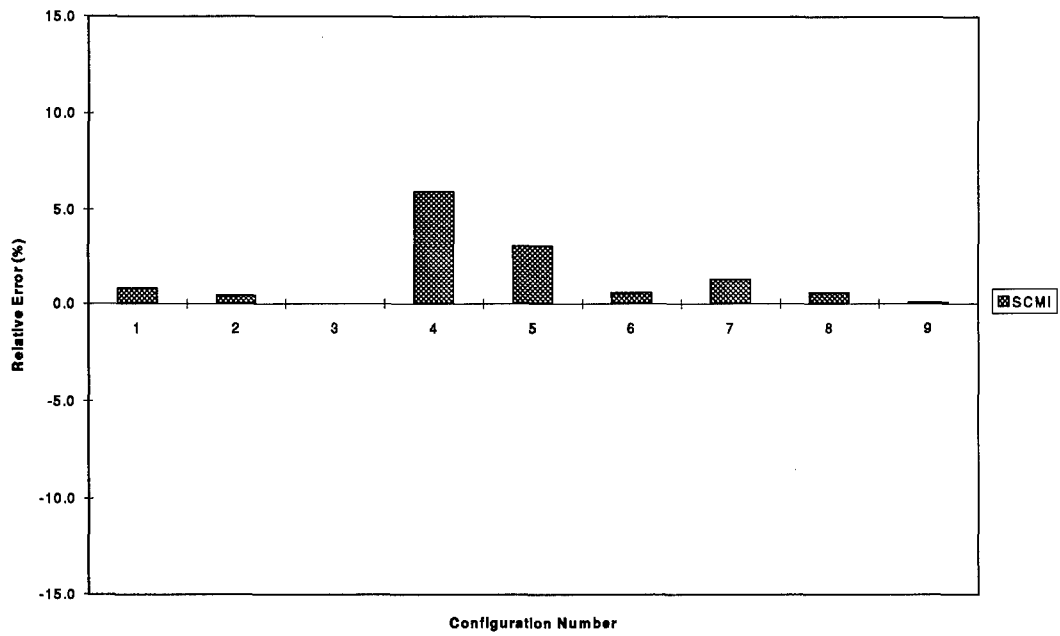


Figure 33. Case Study 1: Expected Queue Length at Station 4, Systems 1-9.

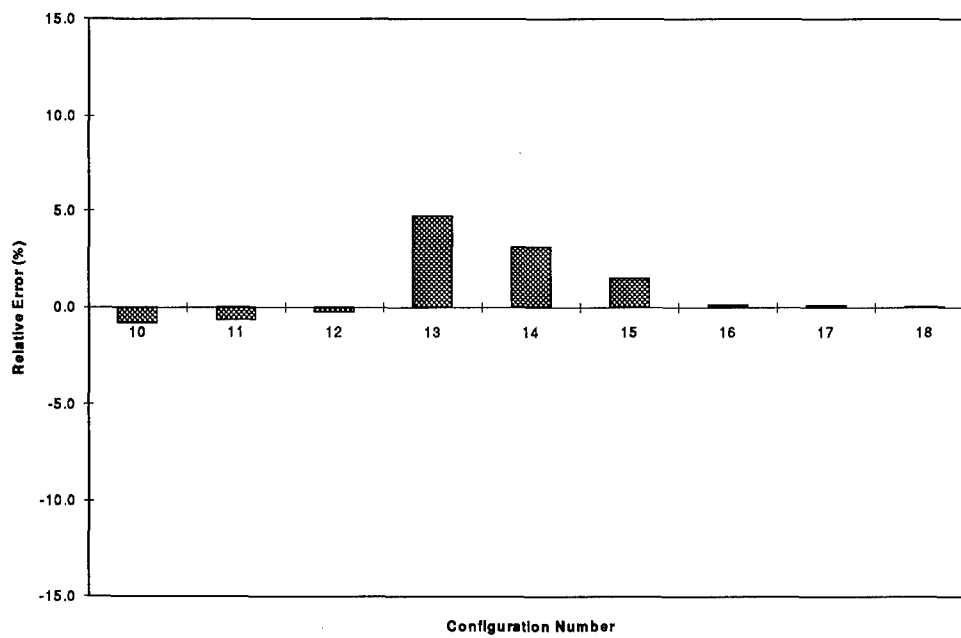


Figure 34. Case Study 1: Expected Queue Length at Station 4, Systems 10-18.

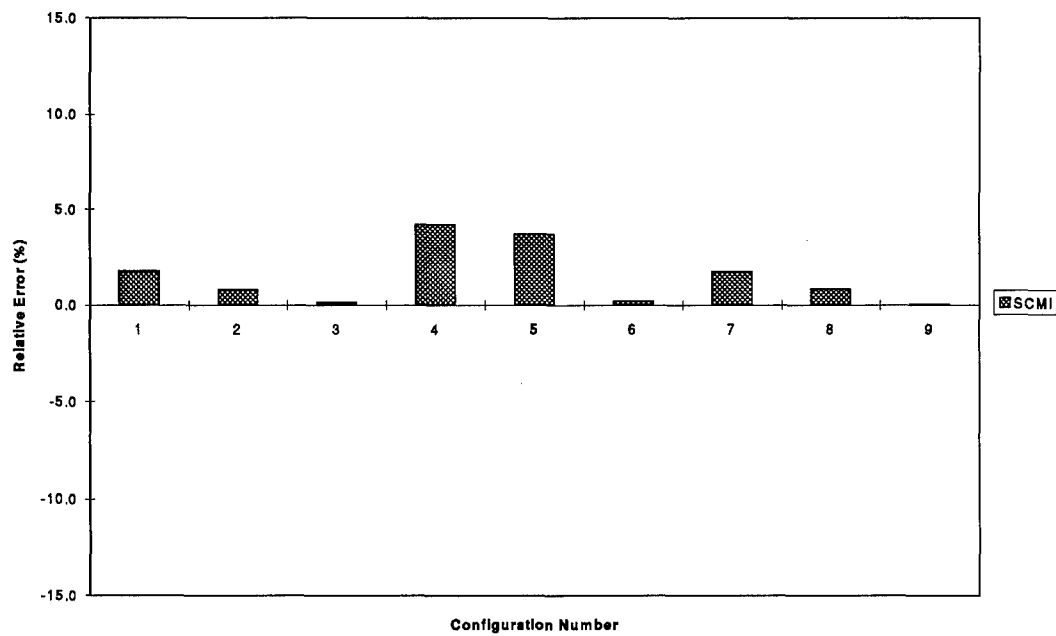


Figure 35. Case Study 1: Expected Queue Length at Station 5, Systems 1-9.

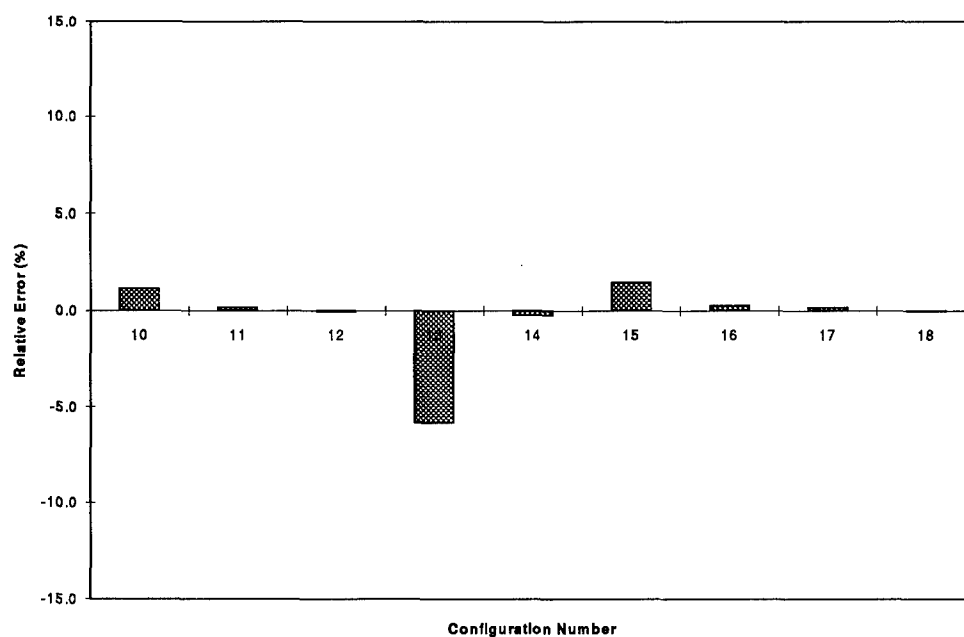


Figure 36. Case Study 1: Expected Queue Length at Station 5, Systems 10-18.

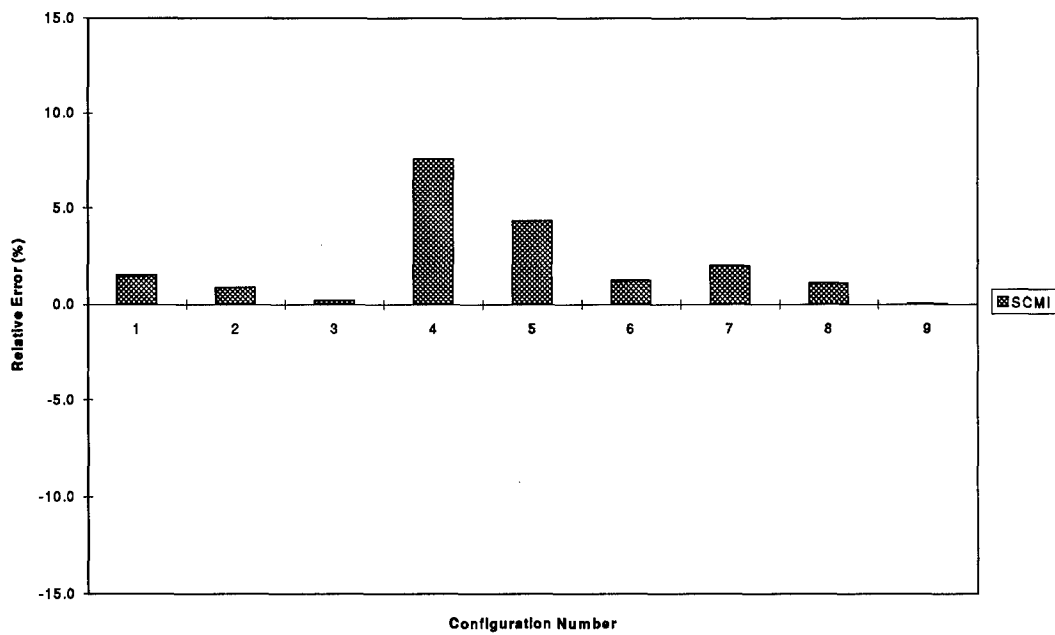


Figure 37. Case Study 1: Expected Queue Length at Station 6, Systems 1-9.

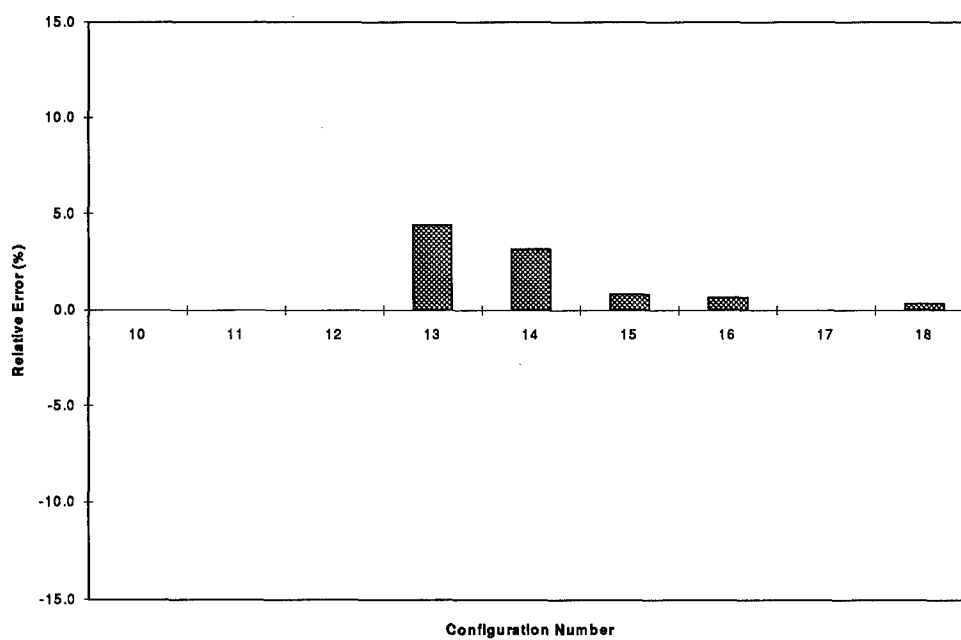


Figure 38. Case Study 1: Expected Queue Length at Station 6, Systems 10-18.

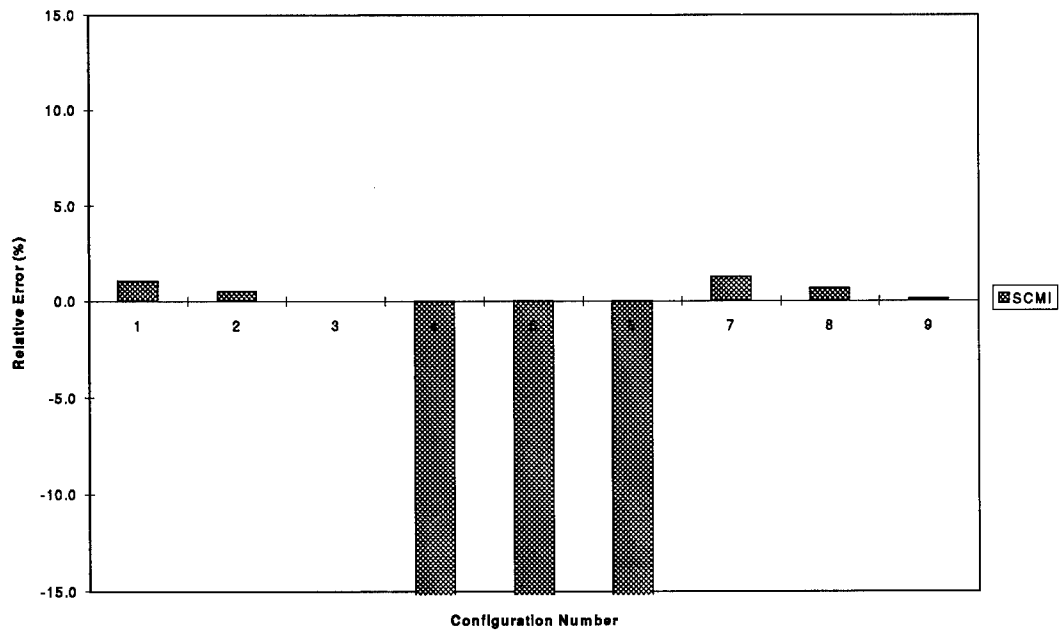


Figure 39. Case Study 1: Expected Queue Length at Station 7, Systems 1-9.

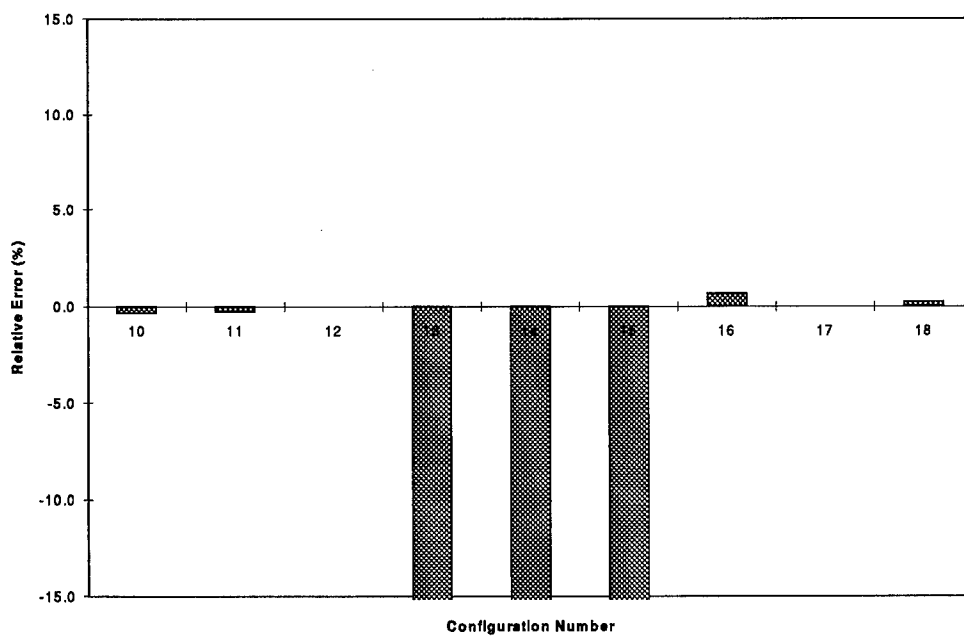


Figure 40. Case Study 1: Expected Queue Length at Station 7, Systems 10-18.

Case Study 2: FJQNs With Nested FJSNs.

Overview. A second numerical study was conducted to examine the performance of SC when applied to nested FJQNs. Based on the performance of SCMI in the previous study, it was chosen as the decomposition method.

As with Case Study 1, eighteen different configurations of the same underlying network topology were studied. The topology is shown in Figure 41. The attributes varied between configurations were the probabilities p_1 , p_2 , p_{11} , and p_{12} , the service time distributions of Stations 5, 7, and 9, and the network population. Response times along the subnetworks in the fork-join structures were intentionally balanced. The specific configurations are enumerated in Table 15. As with Case Study 1, the measure of merit was relative error, and the default stopping strategy was used for Marie's method.

Table 15. Case Study 2: Representative System Configurations.

No	N	Stn 5	Stn 7	Stn 9	$(p_1, p_2, p_{11}, p_{12})$
1	5	2-Erlang(0.5)	2-Erlang(0.5)	2-Erlang(1.0)	(0.5, 0.5, 0.5, 0.5)
2	10				
3	20				
4	5	2-Cox(0.25, 2.5, 0.1)	2-Cox(0.25, 2.5, 0.1)	2-Cox(0.25, 2.5, 0.3)	
5	10				
6	20				
7	5	Exponential(0.5)	Exponential(0.5)	Exponential(1.0)	
8	10				
9	20				
10	5	2-Erlang(0.5)	2-Erlang(0.5)	2-Erlang(0.5)	(0.9, 0.1, 0.9, 0.1)
11	10				
12	20				
13	5	2-Cox(0.25, 2.5, 0.1)	2-Cox(0.25, 2.5, 0.1)	2-Cox(0.25, 2.5, 0.3)	
14	10				
15	20				
16	5	Exponential(0.5)	Exponential(0.5)	Exponential(1.0)	
17	10				
18	20				

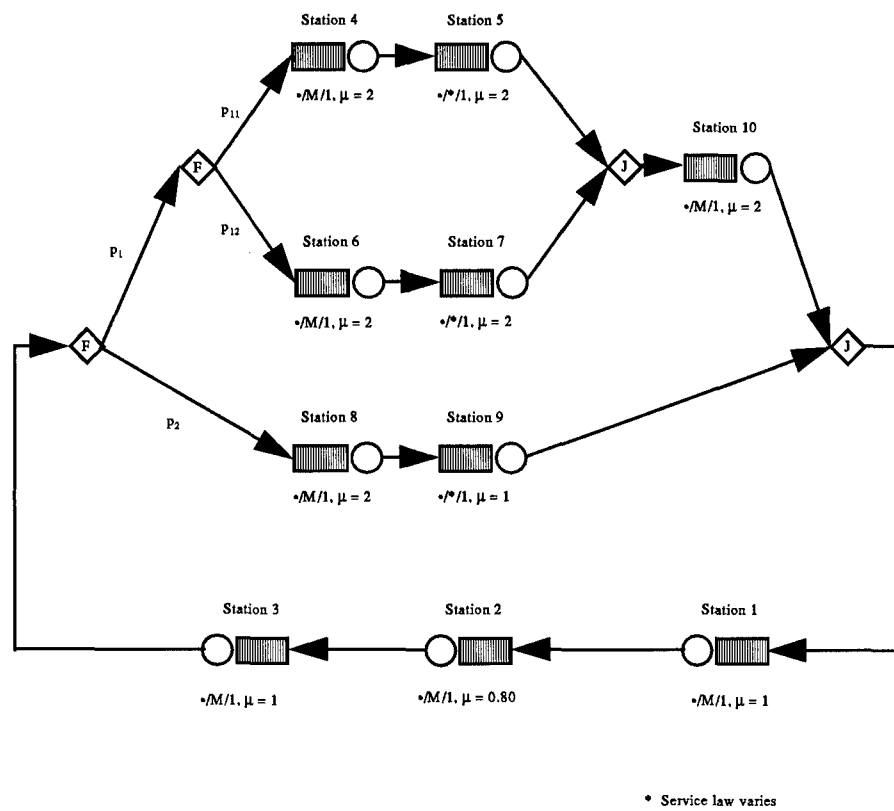


Figure 41. Basic Network Topology, Case Study 2 Representative Systems.

Results. Figure 42 shows the relative error in the expected throughput at Station 1 for each system; relative errors in expected queue lengths are given in Figures 43 to 52. It is clear from the figures that SCMI produced good overall results for the sample networks. Station 9 queue length accuracy suffered for networks where that station had a high-variance Coxian service law (Systems 4–6 and 13–15).

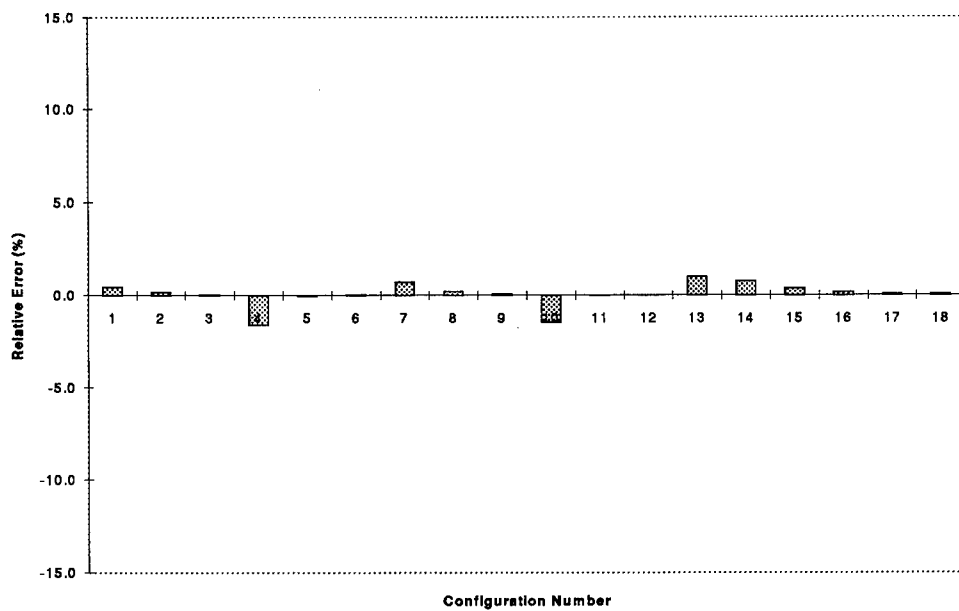


Figure 42. Case Study 2: Expected Throughput at Station 1.

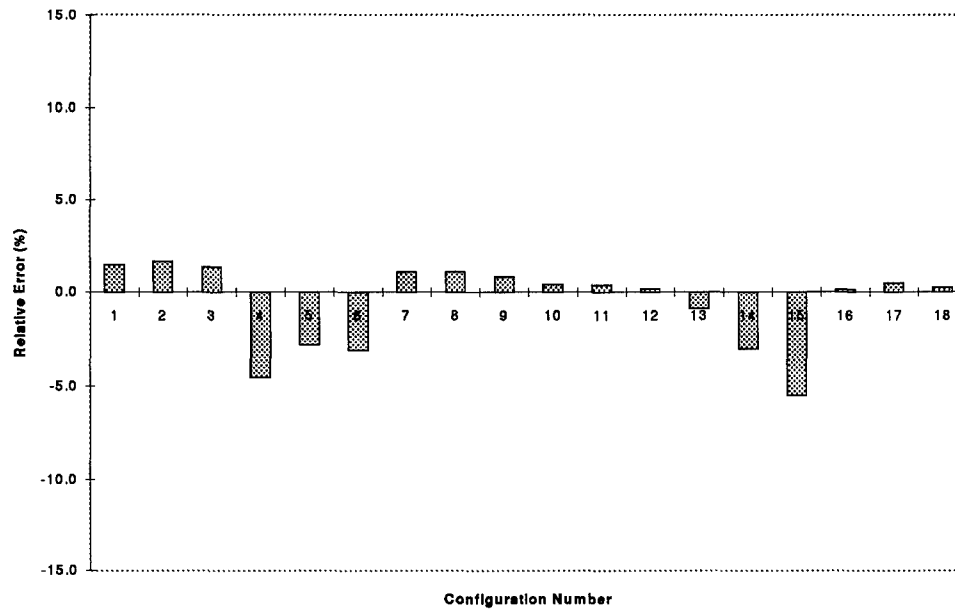


Figure 43. Case Study 2: Expected Queue Length at Station 1.

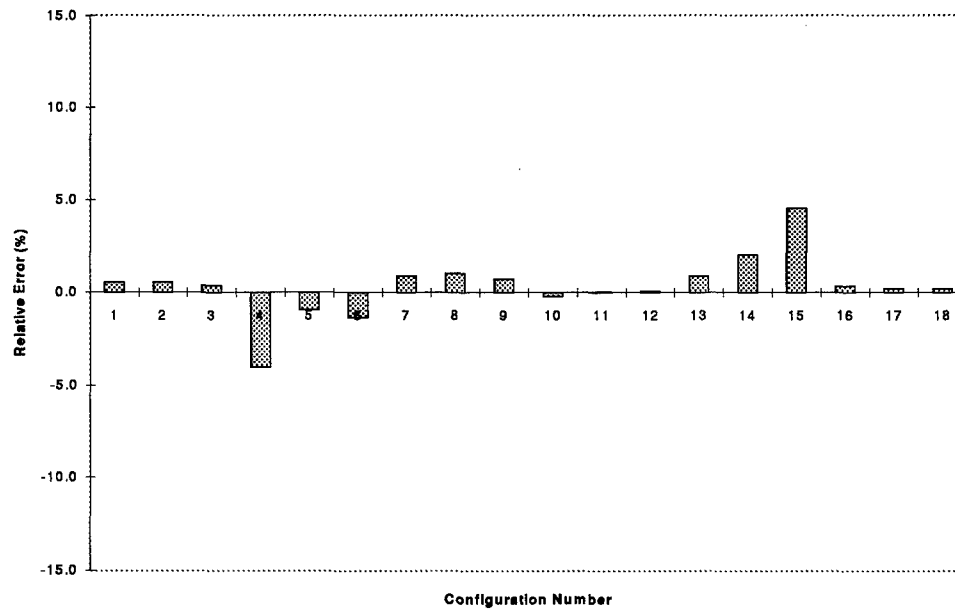


Figure 44. Case Study 2: Expected Queue Length at Station 2.

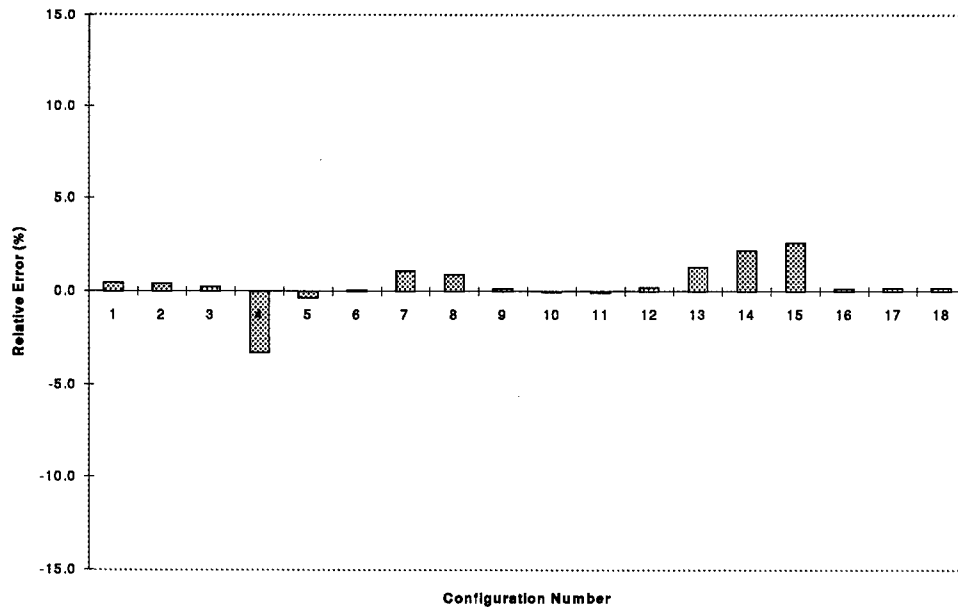


Figure 45. Case Study 2: Expected Queue Length at Station 3.

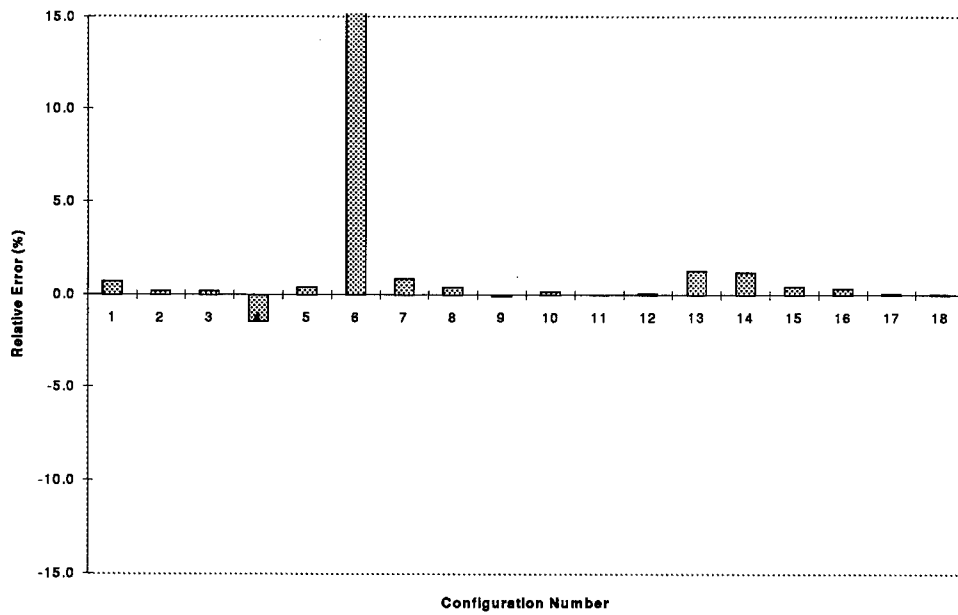


Figure 46. Case Study 2: Expected Queue Length at Station 4.

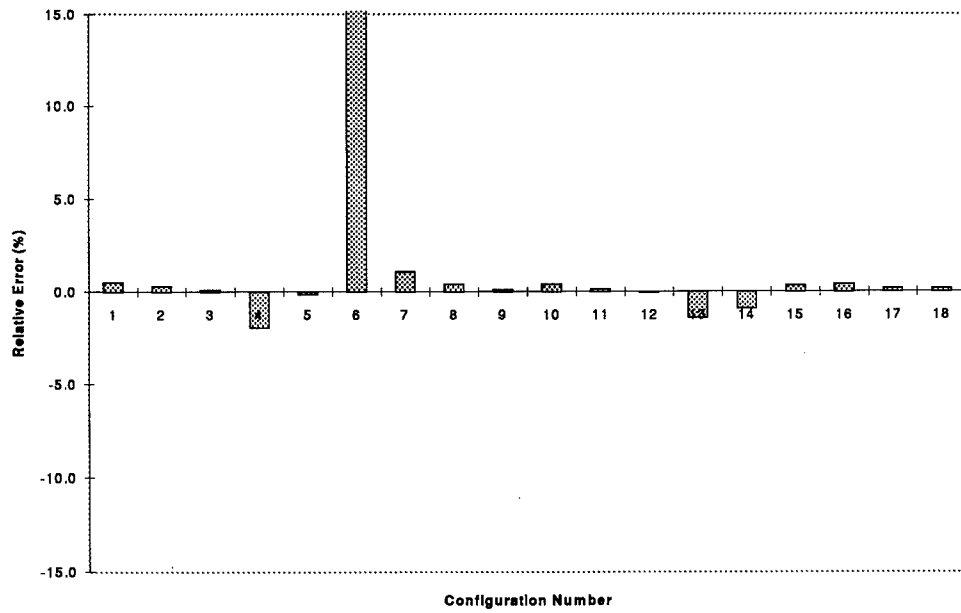


Figure 47. Case Study 2: Expected Queue Length at Station 5.

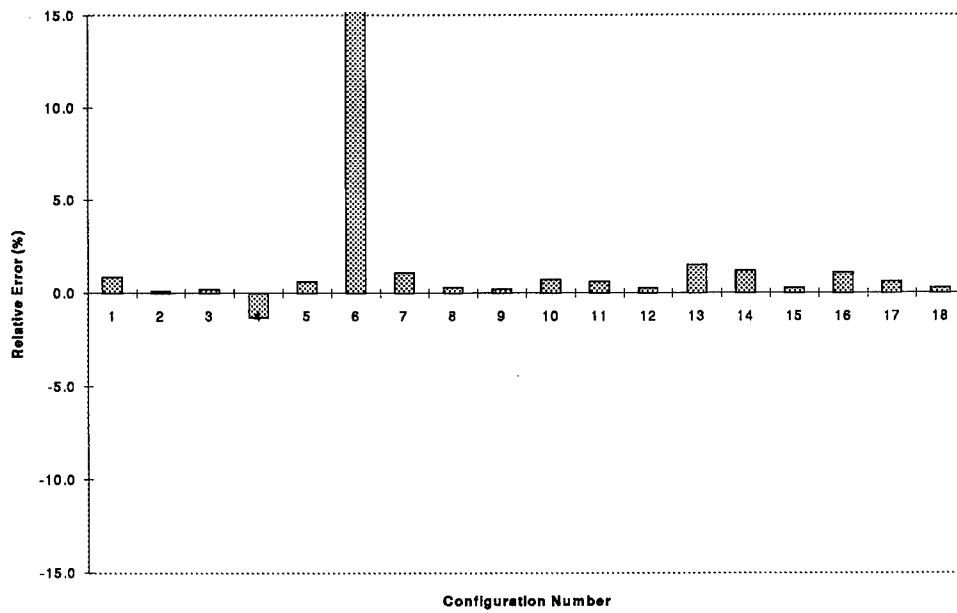


Figure 48. Case Study 2: Expected Queue Length at Station 6.

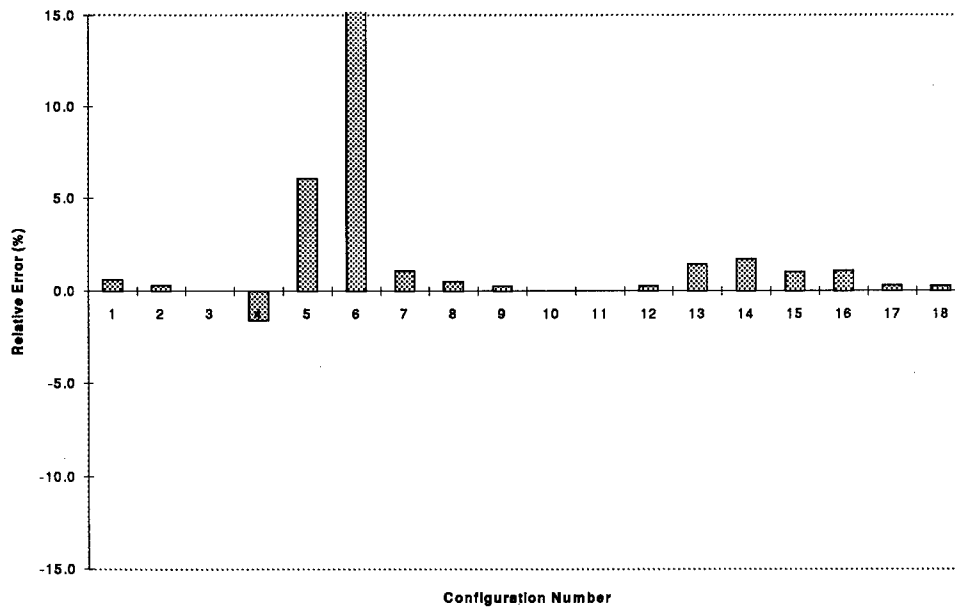


Figure 49. Case Study 2: Expected Queue Length at Station 7.

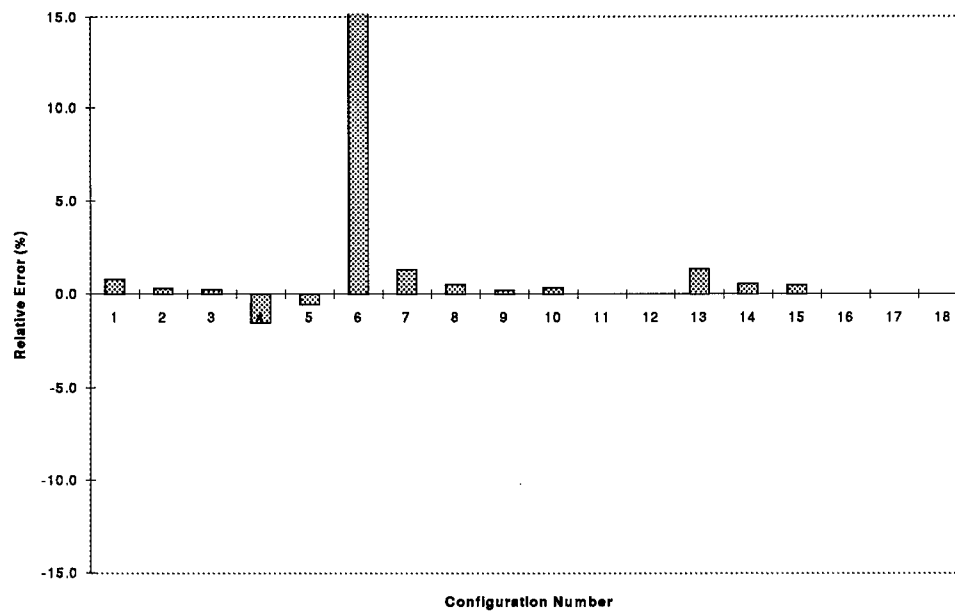


Figure 50. Case Study 2: Expected Queue Length at Station 8.

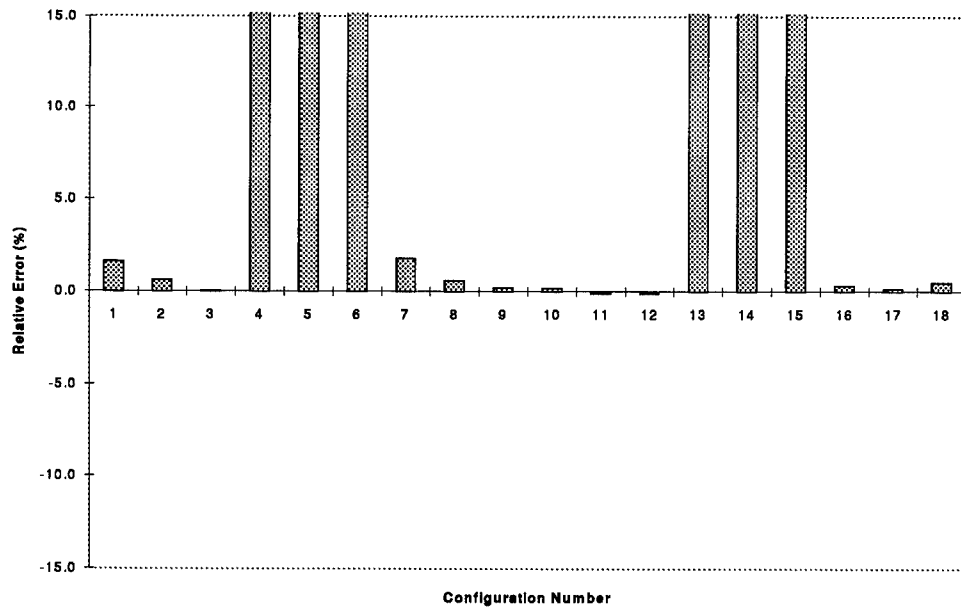


Figure 51. Case Study 2: Expected Queue Length at Station 9.

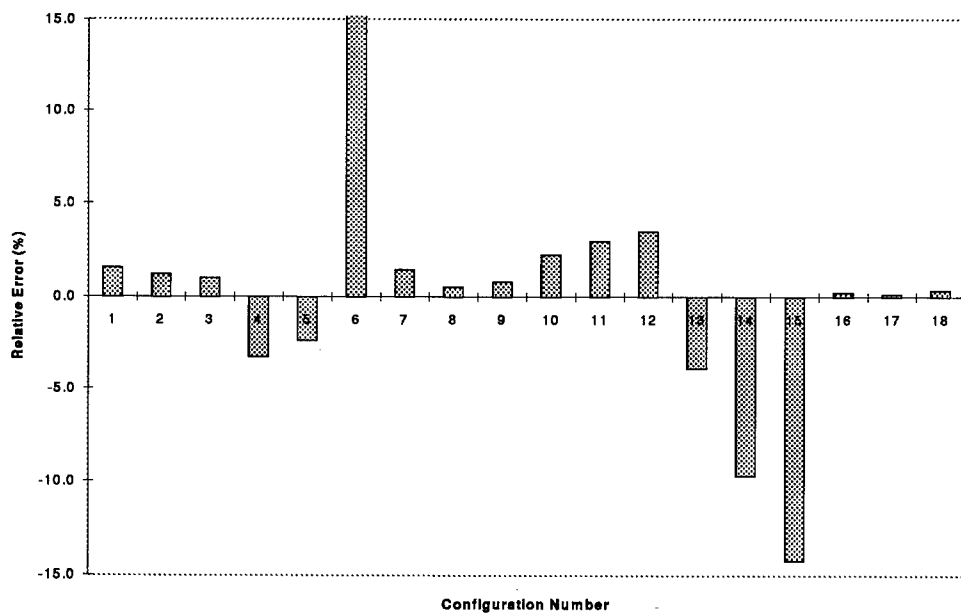


Figure 52. Case Study 2: Expected Queue Length at Station 10.

Special note should be made of the convergence behavior of the nested fixed-point algorithms. The baseline tolerances were set at 10^{-4} at each level. However, most systems required manual adjustment of the tolerance to speed convergence (see Table 16 for the selected tolerance values). In some cases, tolerances had to be raised as high as $o(10^1)$, particularly for higher population levels. It is interesting to note, however, that in all but one case (System 6) the looseness of the tolerances did not seem to directly affect the accuracy of the solutions. Recall that the large relative errors in queue lengths for Systems 4–6 and 13–15 were observed in both numerical studies, and are most likely due to linearizations inherent in SCMI. Since the requirement to adjust the tolerances seemed to be driven by population level, the convergence behavior observed may be due to large relative changes in small service rates (typically found when the station population approaches N). The corresponding absolute changes in the service rates clearly had no practical significance, since the accuracy of the approximate performance measures was not affected by the tolerance selections. Since this convergence behavior was not observed in Case Study 1, it is most likely an effect of nesting the fixed point algorithms, and may be due to a loss of precision. This situation could possibly be rectified by refinements to the stopping test that consider both relative and absolute change in the service rate vectors.

System 6 was a pathological case in that none of the expected queue lengths of Queues 4–10 were within ten percent of truth. This is most likely due to the fact that overall convergence could not be attained without setting the tolerance for the second level fixed-point algorithm at a level that allowed convergence after one pass (see Table 16). Thus, the appropriate elements of the solution vector were not being properly updated.

Conclusion

While the representative systems are hardly a comprehensive picture of reality, the results of the numerical study suggest SC is a useful, and often highly accurate,

Table 16. Case Study 2: Convergence Summary.

Cfg	Maximum Iterations			Selected Tolerance		
	Level 1	Level 2	Level 3	Level 1	Level 2	Level 3
1	2	5	4	10^{-4}	10^{-4}	10^{-4}
2	2	3	3	10^{-4}	10.0	3.0
3	2	5	25	10^{-2}	20.0	10.0
4	2	5	5	10^{-4}	10^{-4}	10^{-4}
5	2	2	18	10^{-4}	15.0	3.0
6	2	1	5	10^{-2}	20.0	10.0
7	2	4	2	10^{-4}	10^{-4}	10^{-4}
8	2	3	2	10^{-4}	25.0	10^{-4}
9	2	2	2	1.0	20.0	10^{-4}
10	2	6	6	10^{-4}	10^{-4}	10^{-4}
11	2	8	2	10^{-4}	5.0	3.0
12	2	4	22	10^{-4}	20.0	7.0
13	2	6	8	10^{-4}	10^{-4}	10^{-4}
14	2	6	32	10^{-4}	7.0	5.0
15	3	8	36	1.5	30.0	15.0
16	2	3	2	10^{-4}	10^{-4}	10^{-4}
17	2	4	2	10^{-4}	10^{-4}	10^{-4}
18	2	3	2	10^{-4}	7.0	10^{-4}

approximation technique for closed FJQNs with probabilistic load patterns. This technique appears equally successful whether or not the network to be analyzed contains nested FJSNs.

In the non-nested case, both SCAE and SCMI produce competitive approximations of expected throughput; this suggests that either approach would be useful if system-level performance measures are desired, particularly those measures that are relatively insensitive to higher moments of the service time distributions. SCMI is clearly the preferred method, since it alone can provide accurate queue lengths for stations inside a fork-join structure.

SCMI seems to suffer no degradation in performance when the network topology contains nested FJSNs. However, larger network populations may necessitate the use of a different stopping criterion for Marie's method.

In some cases, SCMI appears sensitive to service laws having coefficient of variation greater than one. Therefore, the method should be used with care when such stations are present.

Recommendations

The following suggestions for additional study would help to further define the applicability and limitations of SC approximation techniques:

1. Study the effect of combining the SC methods with the class aggregation technique of Baynat and Dallery (see Chapter II).
2. Explore the reasons for the failure of SCAE and SCME to accurately approximate expected queue lengths inside a fork-join structure.
3. Determine the reasons why internal feedback outperforms external feedback for Marie's method but not for aggregation.

4. Explore the use of alternative stopping tests to eliminate convergence problems caused by nesting Marie's method algorithms.

The decomposition strategy described in this chapter can be used to analyze a wide class of queuing networks. This class of models can be applied in many practical situations. Some suggested application areas, together with specific examples, are as follows:

1. Computer performance evaluation: nested parallel algorithm performance, multi-nomial job splitting.
2. Micro-level production models: generalized kanban assembly/disassembly systems.
3. Aggregate production models: multi-plant order fulfillment systems.
4. Ground transportation flow models: ground transportation hub flow, airfield processing flow.

The AAM is an example of the fourth class of applications; its use will be demonstrated in the next chapter.

VI. Analytical Airfield Model Demonstration

Introduction

Based on the material presented in the previous four chapters, we now have a strategy for analyzing the Analytical Airfield Model (AAM) presented in Chapter I. This chapter demonstrates the effectiveness of this approach through a small numerical study of a particular realization of the AAM. Baseline numerical results are presented, together with rudimentary sensitivity analysis. A discussion of these results is presented, including an error analysis. The chapter concludes with recommendations for further research.

Implementing the Analytical Airfield Model

The AAM configuration used in the numerical study was based on a scenario suggested by AMC [106]. The following assumptions were used:

1. The effect of passenger loading/unloading (Stations 3 and 10 in Figure 2) was negligible.
2. Unscheduled maintenance (Station 4 in Figure 2) was not differentiated from scheduled maintenance in the processing flow.
3. There were five types of aircraft used in the scenario, each with a fixed cargo load. The cargo load was either removed from a fully loaded aircraft or loaded onto an empty aircraft during ground processing.
4. Each type of aircraft had a fixed refueling requirement that depended only on its type. Whether or not it required this fuel load depended on its mission.
5. Each aircraft type had a fixed scheduled ground delay.
6. There were five wide-body parking spots, which could accomodate the equivalent of eight narrow-body aircraft.

The arrival stream data were pre-processed to determine the proportion of each type of aircraft, the proportion of aircraft needing fuel, and the mean and variance of the interarrival times. Deterministic service times for all ground processing tasks except refueling were provided by AMC; for cargo processing, these were provided by aircraft type. Pump rates, fuel truck travel times, and fuel line connect/disconnect times were provided so that refueling time could be determined by aircraft type; hydrant and truck pump rates were aggregated by the proportion of each resource at the airfield. Since the AAM can only accommodate a single class of aircraft, aircraft types were aggregated by linearly combining aircraft-dependent mean service times using the proportion of aircraft types obtained during pre-processing. The aggregate service laws assumed for each station are given in Table 17.

Table 17. Analytical Airfield Model Station Descriptions.

Station Number	Activity Description	Number of Servers	Service Discipline	Distribution Type	Visit Prob
0	Interarrival Time	1	FCFS	2-Coxian(3.79,0.65,0.41)	1
1	Landing	1	FCFS	2-Erlang(0.033)	1
2	Taxi/Park	8	Delay	2-Erlang(0.125)	1
3	Scheduled Maintenance (not concurrent with refueling)	8	Delay	2-Erlang(0.083)	1
4	Refuel	6	FCFS	2-Erlang(0.983)	0.47
5	Liquid oxygen servicing	8	Delay	2-Erlang(0.45)	0.47
6	Scheduled Maintenance (concurrent with refueling)	8	Delay	2-Erlang(0.5)	1
7	Cargo off/on	3	FCFS	2-Erlang(0.946)	1
8	Standard Ground Delay	8	Delay	2.34 (Deterministic)	1
9	Backout/Taxi	8	Delay	2-Erlang(0.125)	1
10	Takeoff	1	FCFS	2-Erlang(0.033)	1

Numerical Study

Overview. The baseline AAM configuration was analyzed using the methods proposed in Chapters III and V. Three performance measures were calculated:

1. The airfield throughput (the average number of aircraft leaving the airfield each hour).
2. The airfield response time (the average number of hours it takes between aircraft arrival and departure).
3. The average number of aircraft on station.

These measures were chosen because they are typically of interest to a mobility analyst (see Chapter I).

Sensitivity analysis was conducted by varying the mean interarrival time, the maximum number of aircraft that could be serviced (cargo and fuel), and the coefficient of variation of the service time distributions. Performance measures were also evaluated for the case when the standard ground delay was ignored to capture the effect of standard ground delay on the performance measures.

Error analysis was performed by comparing performance measures calculated with the AAM to a simulation point estimate whose 95 percent confidence interval half-width was less than or equal to 10^{-2} .

Results. Analysis of the baseline scenario showed that, on average, 1.1 aircraft departed the airfield per hour. A typical departing aircraft had spent 2.8 hours on station; on average, 3.0 aircraft were in the airfield flow at any given time. The results of the numerical study are presented in detail in Tables 39 through 55 in Appendix D.

The effect of decreasing the mean interarrival time is illustrated in Figures 54, 55, and 56. Deterministic response time estimates are included in Figure 55 for com-

parison purposes.¹ Clearly, all performance measures increased as mean interarrival time decreased; this makes sense for a capacitated queuing network. Variations in interarrival time had less of an effect on response time than on aircraft on station; this implies that the arrival rate could be increased to drive up throughput provided that the fraction of occupied parking spaces need not be minimized. Further, the effect of ignoring the standard ground delay did not provide much of an increase in throughput even at low mean interarrival times, but it did contribute to large decreases in response time and aircraft on station; ground time could be omitted if these latter two performance measures need to be minimized.

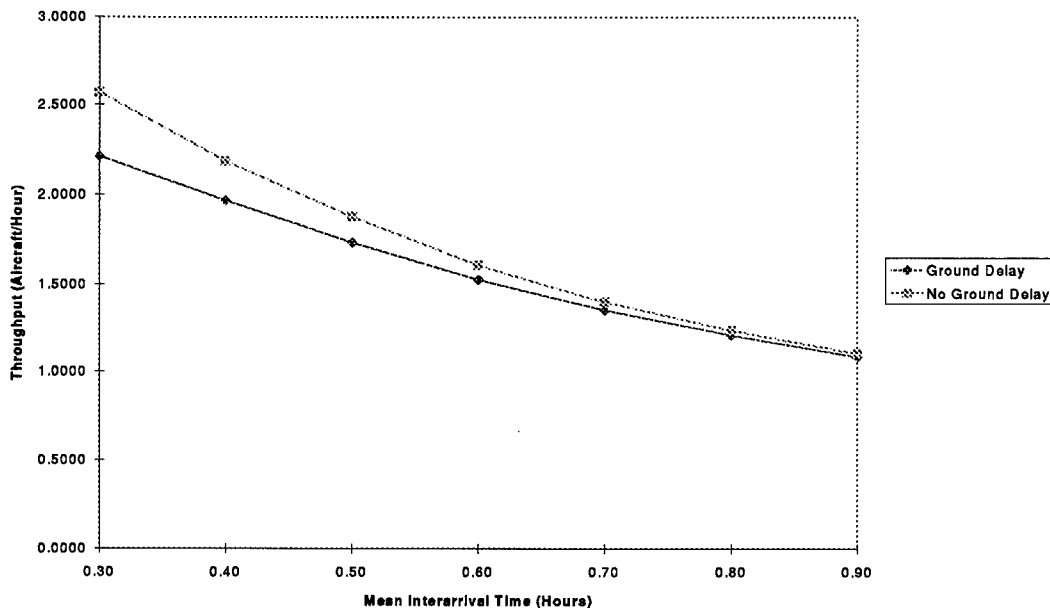


Figure 54. Effect of Mean Interarrival Time on Airfield Throughput.

¹This estimate was calculated by summing the expected response times at each station of the network except Station 0. The mean response time of a fork-join network structure was found by conditioning on the probability of visiting a particular set of forks; for each set visited, the mean response time was taken as the maximum sum of means along the forks.

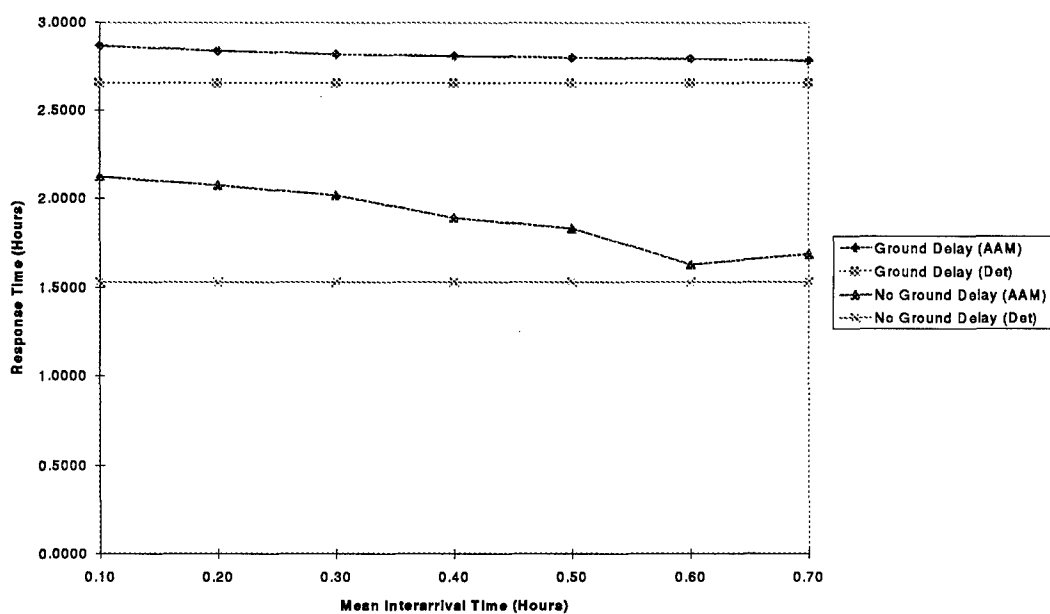


Figure 55. Effect of Mean Interarrival Time on Airfield Response Time.

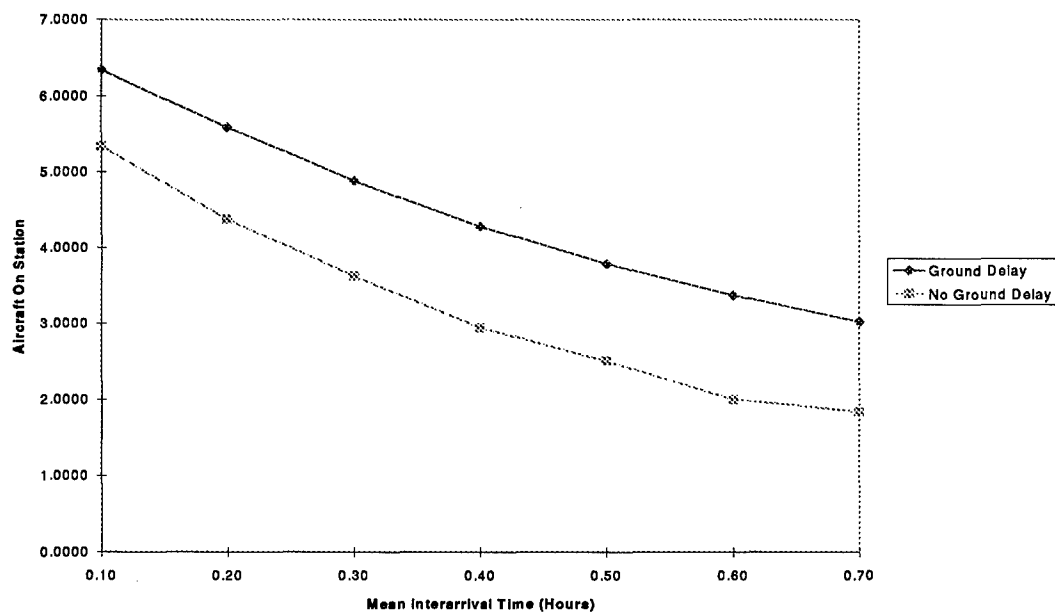


Figure 56. Effect of Mean Interarrival Time on Aircraft on Station.

Figures 57, 58, and 59 capture the effect of varying the maximum number of aircraft that can be serviced for fuel or processed for cargo. The tightly superimposed lines on each graph (each of which reflects a different number of refueling servers, from one to eight) imply that adjusting the maximum number of aircraft that could be serviced for fuel had no significant impact on the values of the performance measures. In contrast, each graph shows the performance measures worsened when resources were adjusted so that only one aircraft's cargo could be processed at a time. This implies that a bottleneck forms at Station 7 if cargo resources are tightly constrained. The figures report results only for the case where ground delay was included; a similar effect was observed when ground delay was ignored.

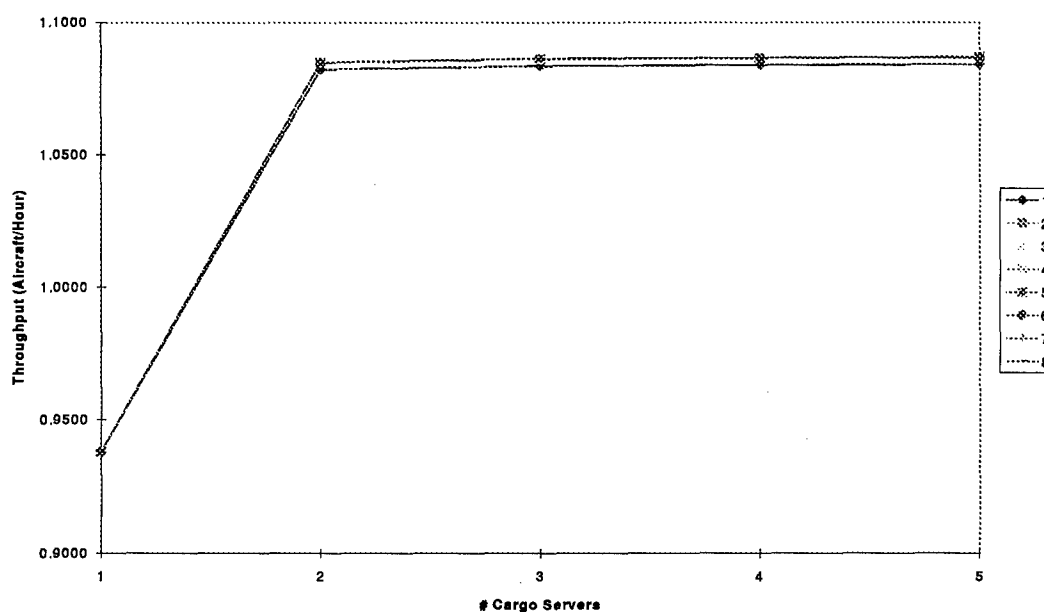


Figure 57. Effect of Constrained Resources on Airfield Throughput.

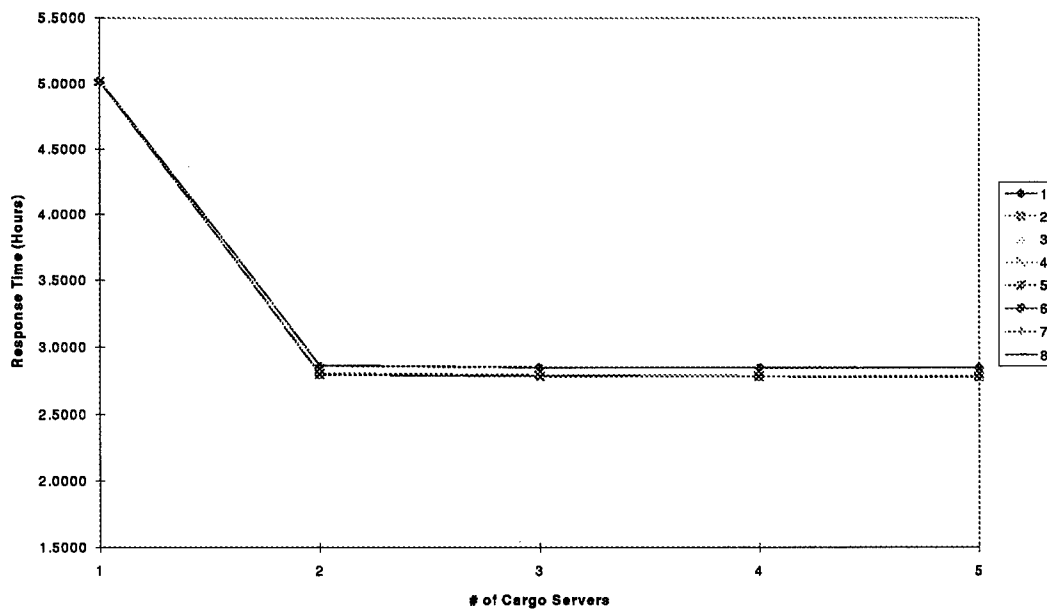


Figure 58. Effect of Constrained Resources on Airfield Response Time.

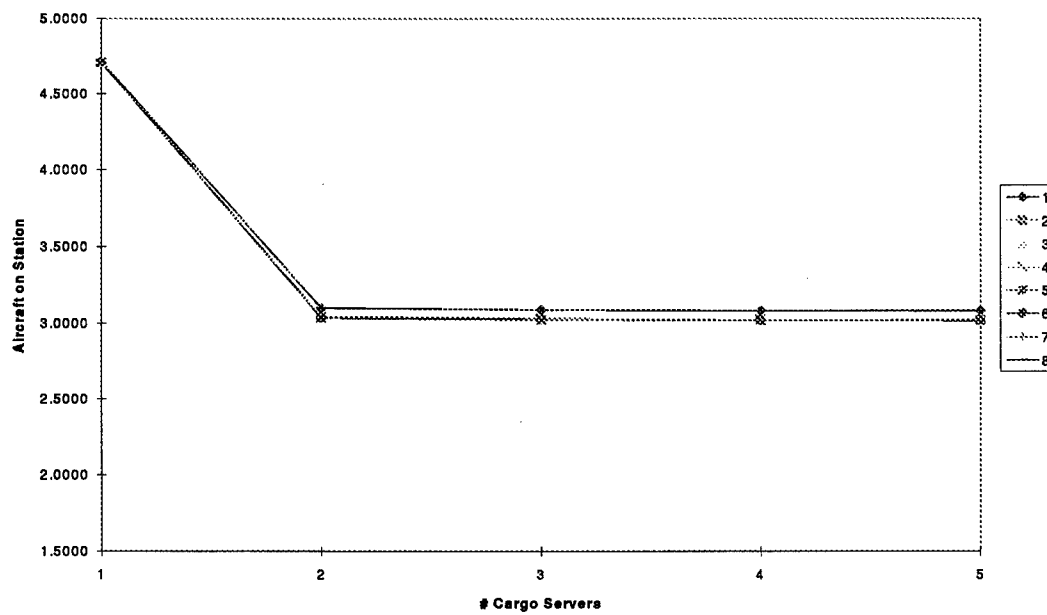


Figure 59. Effect of Constrained Resources on Aircraft on Station.

Increasing the coefficient of variation from 0.5 to 1.0 at each service station (except for the arrival process) had little impact on the performance measures, regardless of the level of constrained resources. The impact was more pronounced in the case where ground delay was ignored, but it was still insignificant from a practical standpoint.

Error Analysis. All but two of the 318 performance measure data points calculated using the analytical model had relative errors of ten percent or less; eighty percent of these were five percent or less. This agrees with the finding in Chapter V that performance measures can be approximated reasonably accurately using the SCMI method.

Further, the response times calculated using the analytical approach were compared to the deterministic response time estimates described above. In every case, the SCMI analysis produced a smaller relative error. The difference in the relative errors was particularly pronounced when ground delay was ignored.

Speed of Execution. The software for the numerical investigation was implemented on a Digital Equipment Corporation Alpha workstation. To achieve the desired tolerance in the simulation point estimates, the typical simulation run took 35 to 45 minutes of elapsed system time. In contrast, virtually all of the analytical software runs produced near-immediate output (within the limitations of output channel bandwidth). This fact, combined with the accuracy of the analytical approximation used, gave the queuing network analysis approach a distinct speed advantage over simulation.

Conclusion and Recommendations

The AAM is a valuable tool for the steady-state analysis of mobility airfield processes. The speed and accuracy with which it can be analyzed make it particularly useful for quick-look studies and sensitivity analysis.

Despite this promising start, further research is needed before such an analytical model can be widely adopted. In particular, the following modeling issues need to be addressed:

1. A variety of approaches to the aggregation of aircraft classes should be studied. If aggregation is not sufficiently accurate, the methods proposed in Chapters III and V need to be adapted to allow their efficient application to multiple-chain queuing networks.
2. A systematic method is needed for modeling the cargo and refueling processes as FCFS multiserver queues, particularly when the servers are heterogeneous.
3. The robustness of the AAM performance measures in the face of a departure from the steady-state assumption needs to be assessed.
4. The effect of the dependence of network capacity on aircraft type needs to be studied further.
5. The feasibility of using AAM-generated performance measures in a system-level mobility model should be explored.
6. The effectiveness of the AAM as a variance-reduction tool should be investigated.

VII. Conclusion

Introduction

This chapter brings this research project to closure. The construction and analysis of the AAM has given rise to several contributions to the literature; these are summarized in the next section. The final section of the chapter consolidates the recommendations for further research that were made in previous chapters.

Statement of Research Contributions

Analysis of Multiserver k -Coxian Queues. A quick, efficient method was identified for solving for the stationary probabilities of the $\lambda(n)/C_k/r/N$ queue. Unpreconditioned Conjugate Gradient Squared was shown to be the method of choice in the context of decomposition using Marie's Method, thus broadening the class of networks where the method is of practical use.

Decomposition of Nested Fork-Join Queuing Networks. A general technique for decomposing nested FJQNs with probabilistic forks was formulated. The approach consists of incorporating feedback loops into the embedded Markov chain of the synchronization station, then using Marie's Method for the network decomposition. Numerical studies showed this strategy to be effective, with less than two percent relative error in the approximate performance measures in most realistic cases.

Isolated Analysis of Mobility Airfield Flow. A single-chain queuing network model of an isolated mobility airfield was formulated. When this network is analyzed using the strategy described above, accurate approximations of airfield performance measures can be obtained in a fraction of the time needed for a simulation study. The proposed modeling approach is especially effective for quick-look studies and sensitivity analysis.

Summary of Recommended Research Topics

Several questions and issues regarding the analysis of nested FJQNs were raised during the course of this study. They are as follows:

1. The effect of combining the SC methods with the class aggregation technique of Baynat and Dallery (see Chapter II) needs further study.
2. The reasons for the failure of SCAE and SCME to accurately approximate expected queue lengths inside a fork-join structure need to be explored.
3. The reasons why internal feedback outperforms external feedback for Marie's method but not for aggregation need to be explored.
4. Alternative stopping tests should be explored in an attempt to eliminate convergence problems caused by nesting Marie's method algorithms.

In addition to these theoretical considerations, the following issues related to the use of the AAM to be considered:

1. A variety of approaches to the aggregation of aircraft classes should be studied. If aggregation is not sufficiently accurate, the methods proposed in Chapters III and V need to be adapted to allow their efficient application to multiple-chain queuing networks.
2. A systematic method is needed for modeling the cargo and refueling processes as FCFS multiserver queues, particularly when the servers are heterogeneous.
3. The robustness of the AAM performance measures in the face of a departure from the steady-state assumption needs to be assessed.
4. The effect of the dependence of network capacity on aircraft type needs to be studied further.

5. The feasibility of using AAM-generated performance measures in a system-level mobility model should be explored.
6. The effectiveness of the AAM as a variance-reduction tool should be investigated.

Appendix A. *Calculating the Stationary Probabilities of a Continuous-Time Markov Chain*

Introduction

The purpose of this appendix is to provide background for the discussion of the computational aspects of continuous-time Markov chains (CTMCs) that occurs in Chapter III. As was seen in that chapter, the efficient and timely calculation of stationary probabilities for these processes is crucial to the effectiveness of certain implementations of Marie's method. The discussion begins by considering the problem of calculating stationary probabilities for a CTMC. The remainder of the appendix reviews methods for solving this problem.

The Stationary Probability Distribution

Definition. Suppose that a certain process can be modeled by a CTMC with n states. For this chain, we define P as the $n \times n$ matrix of instantaneous state transition probabilities, and Q as the $n \times n$ matrix of instantaneous transition rates. If the chain is homogeneous and irreducible, it follows that we can find the $n \times 1$ vector π of stationary probabilities for the chain by solving the following system [116:26–30]:

$$\begin{aligned}\pi^T P &= \pi^T \\ \|\pi\|_1 &= 1\end{aligned}\tag{20}$$

It can be shown (see [116]) that the solution to System (20) is unique, and that P is guaranteed a unit eigenvalue (which is, in fact, the eigenvalue with the largest modulus). Therefore, π is the unique normalized left eigenvector associated with the unit eigenvalue of P , and System (20) is really an eigenvector problem.

For ease of numerical implementation, System (20) is often reformulated in terms of Q . To do this, we first note that $P = Q\Delta t + I$ for some sufficiently small time slice Δt [116:31]. If we apply this fact, and take the transpose of both sides, we have the following equivalent system:

$$\begin{aligned} Q^T \pi &= 0 \\ \|\pi\|_1 &= 1 \end{aligned} \tag{21}$$

The transposed form of the problem is easier to implement and results in more stable execution of certain solution algorithms; see Stewart's detailed discussion in Reference [116:77]. In subsequent discussion, we will use the phrase *stationary probability problem* to refer interchangeably to Systems (20) or (21).

Problem Stability. The success of many numerical solution methods depends on the conditioning of the problem to be solved (i.e., the sensitivity of the solution to small perturbations in the coefficient matrix). Since Q^T is singular, it is impossible to discuss conditioning of the linear system (21) without introducing the concept of the generalized inverse, which is out of the scope of this appendix. However, we can discuss problem stability in terms of the related eigenvalue problem of P (System (20)).

Let π and ϕ be left and right eigenvectors associated with the unit eigenvalue of P . Let \mathcal{B} be the Jordan basis for P , and define \mathcal{B}' as the subspace spanned by $\mathcal{B} - \{\pi\}$. It can be shown [34:15] that the condition number of the unit eigenvalue of P is

$$\kappa_1 = \frac{1}{\min_{y \in \mathcal{B}'} \|\pi - y\|} = \frac{1}{\sin \theta} \tag{22}$$

where θ is the angle between π and ϕ . We know that $\phi = \tilde{e} = (1, 1, \dots, 1)^T$ [116:26–30] Also, the elements of π are clearly nonnegative. Using these facts, it follows

that

$$\kappa_1 = \frac{1}{\sin \theta} = \frac{\|\phi\|_2 \|\pi\|_2}{|(\phi, \pi)|} = \frac{\sqrt{n} \|\pi\|_2}{|(\phi, \pi)|} = \frac{\sqrt{n} \|\pi\|_2}{\|\pi\|_1} \leq \sqrt{n}$$

Therefore, the unit eigenvalue is itself reasonably well-conditioned.

This happy circumstance does not necessarily extend to the conditioning of the normalized left eigenvector π . It follows from a result proven by Chatelin [34:17] that the condition number of π is bounded below:

$$\kappa_\pi \geq \min_{\lambda \in \{\Lambda(P) - \{1\}\}} \frac{1}{|\lambda - 1|} \quad (23)$$

where $\Lambda(P)$ is the set of all unique eigenvalues of P . Equation (23) implies that if P has eigenvalues clustered near the unit eigenvalue, π will be poorly conditioned. A practical effect of ill conditioning is that it adversely affects the accuracy and convergence properties of numerical solution algorithms.

Algebraic Solution of the Stationary Probability Problem

In this section, we consider methods for solving the $n \times n$ eigenvector problem

$$Ax = \lambda x \quad (24)$$

and its equivalent linear system

$$A^*x = (A - \lambda I)x = 0 \quad (25)$$

We will assume that A is nonsymmetric and is not definite. We can approach the task of calculating an eigenvector corresponding to λ using either a direct or an iterative method, since both approaches are equally effective. Accordingly, we will describe methods in both categories. The section ends with a consideration of the application of these techniques to the stationary probability problem.

Direct Solution Methods for Linear Systems. When the CTMC has a small number of states, System (25) is often solved by some *direct method*. This is usually an implementation of the well-known Gaussian elimination procedure, which is defined by the relation

$$a_{ij}^{(s+1)} = a_{ij}^{(s)} - \frac{a_{is}^{(s)}}{a_{ss}^{(s)}} a_{sj}^{(s)}, \quad s+1 \leq i \leq n, \quad s+1 \leq j \leq n \quad (26)$$

where s is the number of the current Gaussian elimination step [6:9].

There are problems with Gaussian elimination that hinder its usefulness, especially for large systems [6:18–21]:

1. It is prone to numerical instability (although implementations have been introduced that reduce numerical error by avoiding subtraction; see, for example, Reference [55]).
2. The sparsity structure of A is destroyed by fill-in (the phenomenon where zero entries in the original matrix become nonzero as the algorithm proceeds). This decreases efficiency and increases memory requirements, particularly in non-sparse implementations, which require storage of the entire matrix.
3. The time requirements for solution quickly become impractical as n increases, because the procedure requires $O(n^3/3)$ floating point operations (flops) to factor A .

To illustrate the limitations of Gaussian elimination, consider a moderately-sized system of order 2048. The solution of this system would require roughly three billion flops and 16 megabytes of memory, not including program overhead (assuming four-bit real data structures).

To alleviate space limitations, Philippe et al. propose a direct algorithm that applies stable Gaussian elimination to large, sparse matrices stored in compact format. This method is specifically intended for Markov chain problems. The authors'

experience shows that the algorithm works well for chains where the order of Q is 2500 states or less, but that memory requirements limit its effectiveness with larger problems [95], [104].

For small Markov Chains, it is often faster and more precise to solve for π using a numerically stable form of Gaussian elimination [104]. As the state space increases, however, it becomes more practical to solve large chains using some other approach.

Iterative Solution Methods. An *iterative method* for solving a system of equations is a procedure that makes repeated improvements to an approximate solution until it is within some tolerance of the true solution [18:1]. There are two classes of iterative methods: *stationary* and *nonstationary*.

Stationary iterative methods have the following general structure:

$$x^{(i)} = Bx^{(i-1)} + c, \quad i \geq 1 \quad (27)$$

where the matrix B and the vector c may depend on A but are independent of i [18:7], [60:5]. These methods are easy to understand and use, but they are usually slower to converge and less efficient than nonstationary methods [18:5].

Nonstationary iterative methods are based on the successive construction of orthogonal vectors. These techniques are sometimes called projection methods. The goal of a projection method is to successively approximate the solution to System (24) or (25) by choosing each approximation in the sequence from a small-dimension subspace (\mathcal{W}) of the solution space such that some function of the iterate is minimized. This choice is typically constrained so that the function of the iterate is orthogonal to a second subspace (\mathcal{L}). The differences between projection methods are determined primarily by the structure of the subspaces \mathcal{W} and \mathcal{L} , and the way in which iterates are constructed from them [95], [104].

An important subset of the projection methods are the *Krylov subspace methods*. These methods are so called because \mathcal{W} is typically chosen to be

$$\mathcal{W} = x^{(0)} + \mathcal{K}_i(r^{(0)}, B) \quad (28)$$

where B depends on A , $r^{(0)}$ is the initial residual, and

$$\mathcal{K}_i(r^{(0)}, B) \equiv \text{span}\{r^{(0)}, Br^{(0)}, B^2r^{(0)}, \dots, B^{(i-1)}r^{(0)}\}$$

is the i th Krylov subspace of $r^{(0)}$ with respect to B [60:11], [93:18]. The subspace \mathcal{L} is typically $\mathcal{K}_i(r^{(0)}, B)$ or some variant thereof [93:18].

Projection methods for eigensolutions. When solving System (24) using a projection method, an approximate eigenvector $x^{(i)}$ is chosen from \mathcal{W} subject to the constraint that $r^{(i)} = Ax^{(i)} - \lambda x^{(i)} \perp \mathcal{L}$. We consider three groups of these methods.

Simultaneous iteration methods are so called because they attempt to successively approximate a set of eigenvectors for A simultaneously. The initial set of vectors is chosen so that they are orthonormal; this orthogonality is maintained throughout the procedure. These methods are primarily of historical interest; they are not as effective as Krylov subspace methods, especially when the moduli of the eigenvalues of A are uniformly distributed [104], [116:186].

One of the oldest Krylov subspace procedures is the *Lanczos Process* [71]. This method uses a three-term recursion technique to generate a nonsymmetric tridiagonal matrix, which is then used to form an approximate eigenvector for A . The projection subspaces are $\mathcal{W} = x^{(0)} + \mathcal{K}_i(r^{(0)}, A)$ and $\mathcal{L} = \mathcal{K}_i(r^{(0)}, A^T)$. Unfortunately, the Lanczos process is not without its problems. First, it requires matrix multiplication by both A and A^T during each iteration, which hampers efficiency [104]. Also, when it is applied to nonsymmetric A , the residuals are orthogonal, but they are

no longer minimized over \mathcal{L} [6:491], [18:21]. Also, the Lanczos process is susceptible to divergence; however, refinements have been proposed that attempt to anticipate convergence problems (see, for example, [116:214–215]).

Arnoldi's Method is related to the Lanczos process. This algorithm takes its name from the fact that it forms an orthonormal basis for \mathcal{W} using modified Gram-Schmidt orthogonalization; this basis generation process was first proposed by Arnoldi [3]. The projection subspaces are defined as $\mathcal{W} = x^{(0)} + \mathcal{K}_i(r^{(0)}, A)$ and $\mathcal{L} = A\mathcal{K}_i(r^{(0)}, A)$. At the i th iteration step, an $i \times i$ upper Hessenberg matrix is formed that is the restriction of A to \mathcal{L} . The eigenvalues of the Hessenberg matrix, particularly those with larger moduli, closely approximate those of A . These eigenvalues are used to construct the appropriate eigenvector of A [6:541–542], [104], [116:190–192].

Projection methods for linear systems. One of the best-known Krylov subspace methods is the *conjugate gradient method* (CG) for solving symmetric positive definite (SPD) systems [6:449–491]. For this method, $\mathcal{W} = x^{(0)} + \mathcal{K}_i(r^{(0)}, A^*)$ and $\mathcal{L} = \mathcal{K}_i(r^{(0)}, A^*)$, where $r^{(0)} = -A^*x^{(0)}$ [93:18]. Unfortunately, CG is unsuitable for nonsymmetric matrices because the sequence of residuals $\{r^{(i)}\}$ cannot be forced to be orthogonal without retaining all of the residual information computed in the course of the solution [18:21]. Iterative methods that accommodate nonsymmetric matrices must confront this difficulty. In the following paragraphs, we describe several of generalizations of the CG algorithm to accommodate nonsymmetric indefinite (or semidefinite) matrices.

Normal equations methods cope with the lack of symmetry by applying CG to the equivalent system

$$(A^*)^T A^* x = (A^*)^T b \quad (29)$$

The rationale behind this approach is that even if A^* is nonsymmetric and indefinite, the normal equation matrix $(A^*)^T A^*$ is SPD. The most efficient of the nor-

mal equations algorithms is Paige and Saunders' LSQR, which uses $\mathcal{W} = x^{(0)} + \mathcal{K}_i((A^*)^T r^{(0)}, (A^*)^T A^*)$, and requires that $(A^*)^T r^{(i)}$ be orthogonal to the following vector space [94]:

$$\mathcal{L} = \mathcal{K}_i((A^*)^T r^{(0)}, (A^*)^T A^*)$$

A drawback of these methods is that their convergence speed depends on the square of the condition number of A^* [18:18]; nevertheless, this limitation has not hindered performance in certain applications [116:219].

The Generalized Minimal Residual (GMRES) method attempts to maintain the orthogonality of the residuals by retaining the entire orthogonal sequence $\{r^{(i)}\}$. In the original implementation of GMRES, due to Saad and Schultz [105], an orthonormal basis for \mathcal{W} is formed using the same modified Gram-Schmidt procedure used in Arnoldi's method, to which it is closely related. As with Arnoldi, $\mathcal{W} = x^{(0)} + \mathcal{K}_i(r^{(0)}, A^*)$ and $\mathcal{L} = A^* \mathcal{K}_i(r^{(0)}, A^*)$. (Walker proposes a modification of GMRES that uses Householder transformations instead of Arnoldi's method; this variant is more numerically stable, but in general is less efficient [130].) Because the entire sequence $\{r^{(i)}\}$ must be stored, GMRES is seldom implemented in its pure form. For practical purposes, the algorithm is restarted after a certain number of iterations, say j ; that is, all information is cleared except $x^{(j)}$, which is used as the new initial solution. This restarted procedure is referred to as GMRES(j). There is no "correct" value for j ; the choice must be made based on computational experience with a certain class of problems [18:18–21].

As the name implies, *Lanczos methods* are based on the Lanczos process described earlier. Like this process, Lanczos methods attempt to retain the orthogonality of the residuals at the expense of minimization. Fletcher's Bi-Conjugate Gradient (Bi-CG) algorithm [50] is the linear-system analog of the Lanczos process; it generates two mutually-orthogonal series of residuals, one with respect to A^* , and the other with respect to $(A^*)^T$. As with Lanczos, $\mathcal{W} = x^{(0)} + \mathcal{K}_i(r^{(0)}, A^*)$, and $\mathcal{L} = \mathcal{K}_i(r^{(0)}, (A^*)^T)$ [93:18]. Because it has the same tendencies to break down as the

Lanczos process, Bi-CG has irregular convergence behavior; the closely related Quasi-Minimal Residual method of Freund and Nachtigal attempts to smooth convergence decomposing A^* in a more stable manner, and by anticipating breakdowns in the underlying process [18:23], [51]. The Conjugate Gradient Squared (CGS) method, first proposed by Sonneveld, converges faster than Bi-CG and improves efficiency by avoiding the use of $(A^*)^T$ [111]. CGS may be thought of as Bi-CG with the residual reduction operator applied twice during each iteration; therefore, CGS should be expected to converge twice as fast as Bi-CG (although this is not uniformly true in practice) [18:26]. For certain problems, CGS is prone to cancellation effects in the updates of the residuals, which affect the stability of convergence [18:26], [125]. To counteract this effect, van der Vorst proposes a variant of CGS called Bi-CGSTAB, which ensures that the residual vector is minimized at least locally [126]; this work is extended to add even more stability by Chan and Szeto [31].

Preconditioned systems. The convergence of an iterative method is strongly dependent on the spectral properties of the matrix A^* . If A^* is ill-conditioned, a method's convergence behavior may become erratic. Therefore, it is common practice to transform the system $A^*x = 0$ into an equivalent system that is better conditioned. This technique is called *preconditioning*. To precondition System 25, a transformation matrix M is derived, and the system is transformed as follows:

$$(M_1^{-1}A^*M_2^{-1})(M_2x) = 0 \quad (30)$$

where $M = M_1M_2$. If M_1 and M_2 are non-trivial, the method is called two-sided preconditioning; otherwise, the new system is said to be left-preconditioned if $M_1 = I$, or right-preconditioned if $M_2 = I$ [18:39–40], [93:12].

Preconditioners generally fall into two classes: *point* and *line* (or *block*). Line preconditioners are useful if A^* has some readily-exploitable block structure, such as block tridiagonality. Otherwise, a point preconditioner is applied [93:12].

The point preconditioning techniques most commonly used in Markov chain applications are based on *incomplete factorization*. The basic form, known as incomplete LU factorization ($ILU(k)$), is a modification of Gaussian elimination that limits or discards off-diagonal fill-in. The value of k determines the level of fill-in allowed. Greater fill-in increases the accuracy of M (i.e., it decreases $\|A^* - M\|$) at the expense of stability and memory [7:39]; see Meijerink and van der Vorst [82] for numerical examples.

For general k , the matrix M takes the form

$$M = (\Delta - L)\Delta^{-1}(\Delta - U) \quad (31)$$

where Δ is a diagonal matrix containing the factorization pivots, L is lower-triangular, and U is upper-triangular [93:13]. For $k = 0$, it may be shown that

$$M = (D - S)D^{-1}(D - T) \quad (32)$$

where diagonal D , lower-triangular S , and upper-triangular T are chosen such that $A^* = D - S - T$ [18:43]. The space complexity of this preconditioner is on the order of the number of nonzero entries in A^* [104].

It is possible to modify the $ILU(k)$ preconditioner so that M has the same row sums as A^* ; this ensures that $M^{-1}A^*$ has a unit eigenvalue [93:14]. This procedure is quite successful when applied to certain classes of problems, particularly in the area of finite element analysis [18:44]. For some types of linear system, however, this modified approach may break down [125].

Other modifications to the ILU procedure center on eliminating elements based on relative magnitude rather than preserving the sparsity structure of A^* . One successful variant of $ILU(k)$ is $ILUTH$, which retains all fill-in entries that exceed a preset tolerance (typically $< 10^{-3}$). A second effective method is $ILUK$, which

retains a predetermined number of the largest entries in a row after each elimination step. (Both methods retain diagonal entries regardless of their magnitude.)

The concepts behind the point ILU(k) preconditioner can be used to construct a block matrix form of the decomposition. Line ILU(k) preconditioners will not be discussed here. The reader is referred to Axelsson [6:260–287] and Oppe et al. [93:15–17] for comprehensive theoretical development.

Incomplete factorization is not the only way to precondition a linear system. One other method commonly encountered is *symmetric successive overrelaxation* (SSOR). The SSOR M matrix is constructed as follows:

$$M = \frac{1}{2-\omega} \left(\frac{1}{\omega} D - S \right) \left(\frac{1}{\omega} D \right)^{-1} \left(\frac{1}{\omega} D - t \right) \quad (33)$$

where D , S , and T are defined above, and ω is an overrelaxation parameter whose optimal value must usually be guessed [93:13].

The trade-off between the computational cost of preconditioning and the anticipated acceleration in convergence must be weighed when applying a preconditioner. This is especially true for preconditioners based on incomplete factorization, which tend to be expensive, especially for large systems [18:39].

Application to Markov chain problems. Projection methods have been shown to be highly effective for solving both System (20) and System (21). Recent comprehensive work in this area has been done by Philippe et al. [95] (see also [104] and [116:121–230]). The authors show that a number of Krylov subspace methods for both eigensolutions and linear systems are effective for medium- to large-sized chains. (In a contemporary article, Krieger and Sczittnick also investigate iterative solution procedures, but they limit their inquiry to stationary methods [67].)

It is axiomatic that there is no single “best” iterative method for solving linear systems (although it can be shown that a superior method can be found within a

particular problem class) [18:37]. This assertion is supported by the computational experience of Philippe et al., who found no optimal method for solving the stationary probability problem for a general CTMCs arising from applications in queueing network analysis [95].

When solving for stationary probabilities, incomplete factorization is usually the preferred preconditioner. This may be due to the fact that an ILU decomposition can be shown to exist for any Q and Q^T by trivially extending results given by Axelsson and Barker [7:42–44]. Philippe et al. found that incomplete factorization outperformed SSOR, especially for large CTMCs; they observed the best performance from ILUTH [95], [104]. Saad finds no intuitive explanation for why there are variations in performance between the types of ILU schemes [104].

As with direct solution algorithms, the size of the problems to which iterative methods can be applied is limited in practice by available memory. However, large Q matrices tend to be sparse, so the use of a sparse matrix storage scheme can allow iterative methods to be used on problems many times the size of those where direct solution is effective. Innovative generation algorithms, such as the one proposed by Knaup that makes use of concepts from tensor algebra, can be used to increase the upper limit on size even further [64]; for example, Knaup showed that his method can be used to get stationary probabilities for sparse Q matrices of order up to 10^7 . This makes iterative methods an attractive alternative for analyzing large-dimension CTMCs.

Solving the Stationary Probability Problem Using Decomposition

Sometimes a Markov chain can be partitioned into a number of subchains, where the magnitudes of the probabilities of transitions between the states in each subchain are large compared to those of transitions between subchains. If such a partition exists, the original chain is said to be *nearly completely decomposable* (NCD), *nearly uncoupled*, or *nearly separable* (see Courtois [36] for rigorous development).

If a Markov chain is NCD, its corresponding system of stationary equations is often ill-conditioned [95]. This causes problems with iterative solution methods. Fortunately, the chain's decomposability can be exploited to construct a smaller, nearly-equivalent chain. The stationary probabilities associated with the new chain can then be used to approximate those of the original chain. This process is called *aggregation/disaggregation* (A/D).

A/D algorithms follow three basic steps:

1. Partition the chain into a set of nearly-independent subchains (ideally, the number of subchains is considerably smaller than n).
2. Compute the stationary probabilities for each subchain.
3. Construct the stationary probabilities of the original chain from the subchain.

In general, this process is not exact, because the subchains are not strictly independent; however, if a Markov chain is NCD, independence almost holds, and the induced error will be small [118].

The comprehensive surveys by Schweitzer [108] and Stewart [116:285–342] provide a solid overview of basic A/D techniques. In one of the more recent developments, Kim and Smith identify a large class of CTMCs for which the A/D algorithm produces exact results [62]. Investigations by Stewart [116:307–331], Stewart and Wu [118], and Touzene [122] suggest that combining A/D with iterative solution schemes works well for NCD chains. Shioyama and Tanaka [110] formulate the partition step as a decision problem, and use a branch-and-bound technique to find the optimal partition.

A/D can be used with any CTMC. However, it is important to remember that this is an approximate method, and that its accuracy depends on the chain's decomposability. If the error that would be induced by applying A/D to a non-NCD chain is unacceptable, some other solution method should be used.

Solving the Stationary Probability Problem Using Recursion

If Q has a block-Hessenberg structure, a recursive solution algorithm can often be used to solve for π . Usually, these approaches require the subblocks of Q to be square matrices of identical dimension. Such methods can be grouped into two categories: general *block-recursive methods* and *matrix-geometric solution algorithms*.

General block-recursive methods are often suggested by the form of the Chapman-Kolmogorov equations for a particular problem. Such general approaches can be an effective way to solve for π , but they must be implemented carefully to guard against numerical instability [116:250]. There also exist extensions of algebraic methods (such as those described previously) to the case where the matrix can be partitioned into submatrices. These block-recursive methods can be quite stable and effective. Gaussian elimination has been extended to block form, as has the stationary Gauss-Seidel iterative method; both are described in Stewart [116:138–142; 234–257]. Obviously, other iterative methods can be extended to the block-matrix case. These methods can also be used as preconditioners, as mentioned earlier [93:12].

The second category of procedures for CTMCs with block-Hessenberg structure are the matrix-geometric solution methods proposed by Neuts [90], [89]. These algorithms are more restrictive than the general block-Hessenberg case, because they require P to have the form

$$P = \begin{bmatrix} B_0 & A_0 & 0 & 0 & 0 & \dots \\ B_1 & A_1 & A_0 & 0 & 0 & \dots \\ B_2 & A_2 & A_1 & A_0 & 0 & \dots \\ B_3 & A_3 & A_2 & A_1 & A_0 & \dots \\ B_4 & A_4 & A_3 & A_2 & A_1 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

where A_i and B_j are square matrices of the same order for all i and j , and A_1 and B_0 are nonsingular. Neuts has developed a set of iterative techniques for solving for π when P has this general form; others have refined his methods. In particular, much attention has been paid to the case where P is block-tridiagonal (i.e., $A_i = B_j = 0$ when $i > 2$ and $j > 1$); such a CTMC is called a quasi-birth-death process (QBD). Quadratically converging algorithms exist for solving these types of problems. See References [90:81–141] and [116:258–283] for details.

Appendix B. *The $\lambda(n)/C_k/r/N$ Queue: Numerical Results for
Generalized k -Erlang Systems*

Table 18. Elapsed Time to Solution (in seconds), No Preconditioner.

Number	GMRES(100)	GMRES(75)	GMRES(50)	CGS	LSQR
24	5.06	3.52	4.01	1.18	4.92
25	25.68	20.62	14.24	1.72	11.88
26	2.37	1.96	2.17	0.68	4.25
27	25.11	11.59	10.65	2.35	14.64
28	29.24	23.98	19.53	4.66	33.52
29	5.11	4.72	5.23	1.35	5.96
30	32.76	18.51	16.12	4.32	31.34
31	94.88	76.09	74.48	10.29	79.68
32	9.33	9.07	10.60	2.58	17.07
33	1.57	1.43	1.55	0.64	4.07
34	11.87	10.59	12.46	3.91	37.54
35	36.32	45.88	47.75	10.95	F
36	86.90	114.04	112.73	24.27	F
37	91.47	98.40	76.82	14.15	163.06
38	173.64	146.39	118.45	39.80	F
39	409.93	436.60	590.52	78.02	F
40	59.71	129.81	117.68	30.70	362.32
41	9.52	8.68	7.75	2.79	24.22
42	44.74	32.88	28.36	6.84	64.88
43	77.07	86.96	70.13	13.69	F
44	10.01	10.15	9.12	5.04	49.55
45	70.04	62.41	73.29	31.87	F
46	173.51	202.56	239.55	70.78	F

F = failed to converge by iteration 2000

Table 19. Elapsed Time to Solution (in seconds), ILU(0) Preconditioner.

Number	GMRES(100)	GMRES(75)	GMRES(50)	CGS	LSQR
24	2.58	2.50	2.33	1.56	2.58
25	4.61	4.30	4.46	2.46	4.32
26	1.55	1.54	1.53	1.28	2.30
27	5.68	5.11	5.06	3.88	6.50
28	12.05	13.46	11.72	8.43	14.88
29	3.21	3.12	3.13	2.60	3.75
30	10.99	11.16	11.11	8.94	14.90
31	27.29	24.13	23.82	18.78	33.61
32	4.44	4.30	4.03	3.25	4.66
33	1.19	1.18	1.19	1.12	1.53
34	11.70	11.78	11.52	10.13	14.01
35	40.72	38.48	35.98	33.97	46.70
36	94.01	83.47	84.40	70.57	97.58
37	68.17	59.90	66.53	61.44	74.24
38	272.17	257.99	251.40	242.62	318.59
39	599.95	570.42	580.82	538.27	639.48
40	204.21	193.38	197.45	197.59	248.49
41	4.45	4.39	4.34	3.43	4.76
42	10.10	10.36	9.87	7.59	11.77
43	39.67	26.48	20.43	14.29	24.12
44	19.66	19.32	19.10	19.67	22.16
45	159.83	162.19	149.12	155.39	176.00
46	483.25	488.77	477.21	442.48	536.45

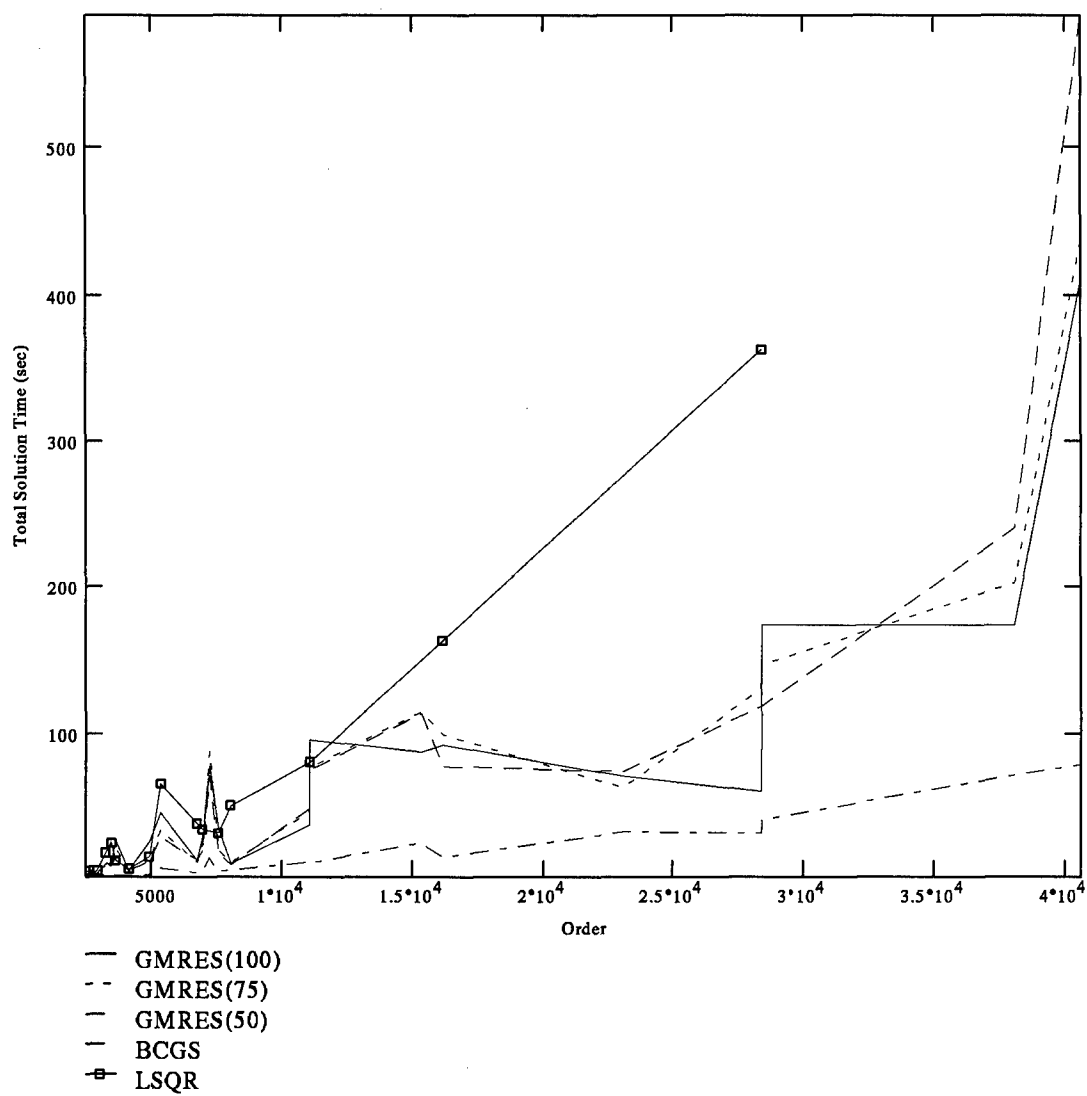


Figure 60. Generalized k -Erlang Systems: Solution Time vs. Order, No Preconditioning.

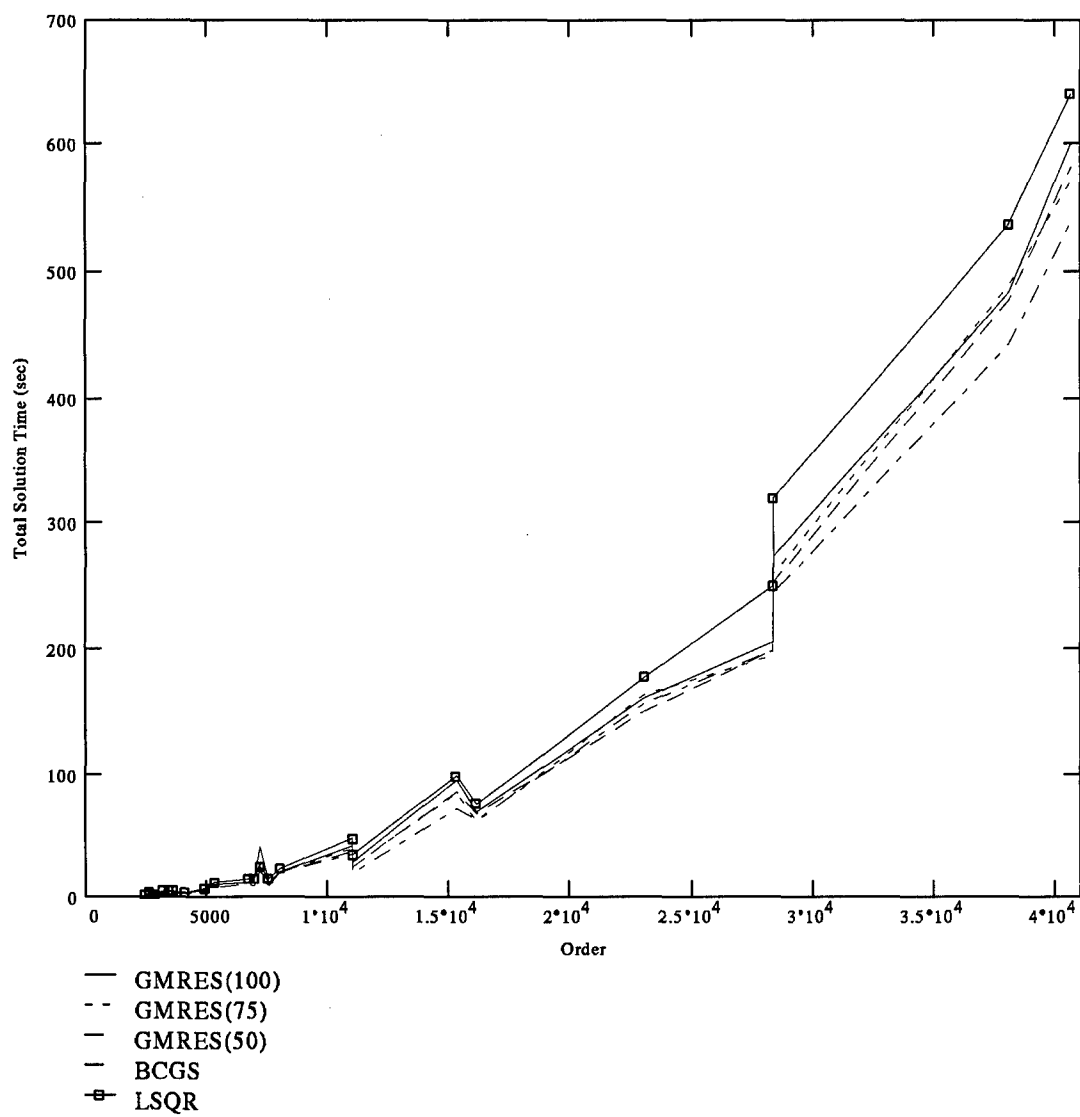


Figure 61. Generalized k -Erlang Systems: Solution Time vs. Order, ILU(0) Pre-conditioner.

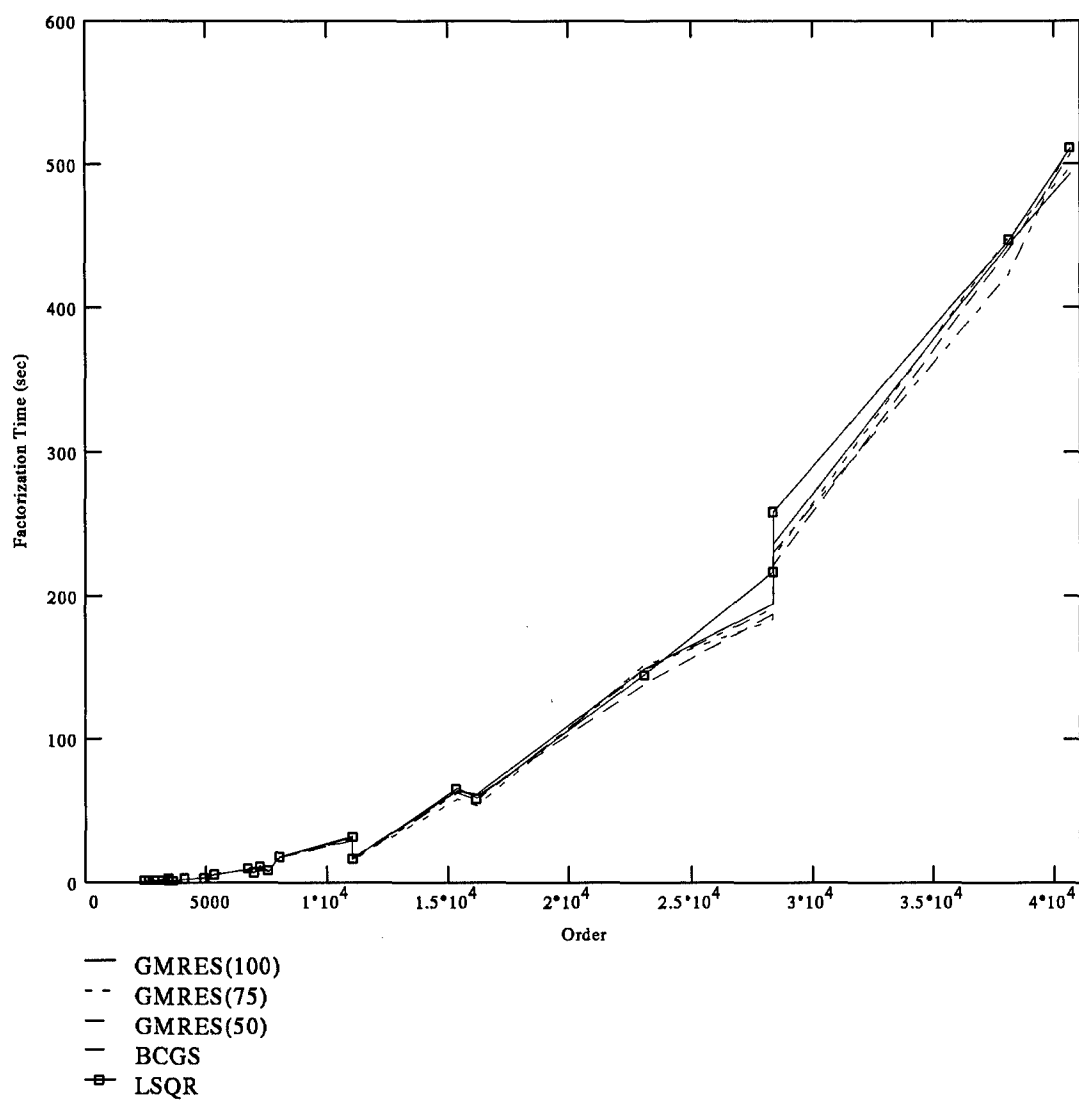


Figure 62. Generalized k -Erlang Systems: Factorization Time vs. Order, ILU(0) Preconditioner.

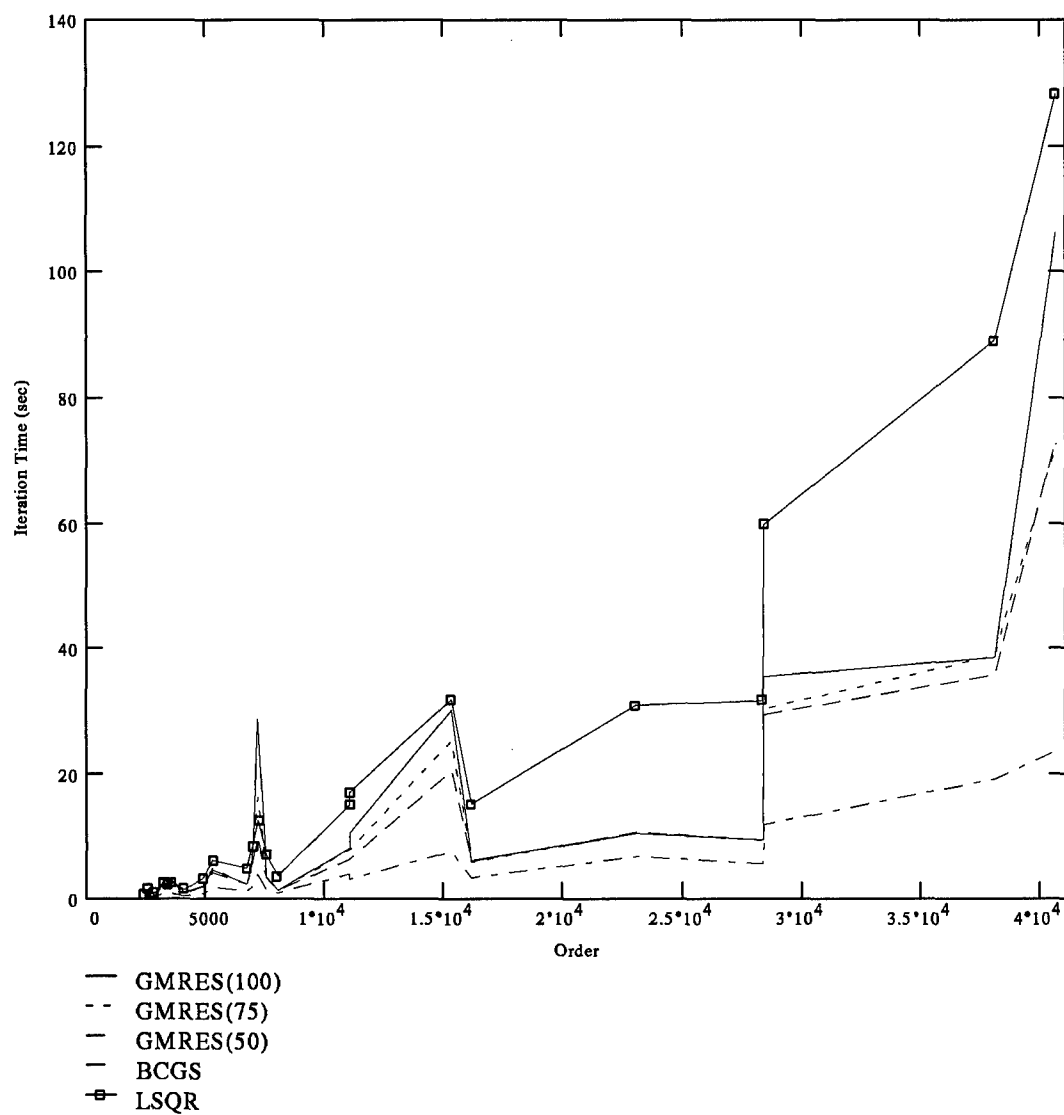


Figure 63. Generalized k -Erlang Systems: Iteration Time vs. Order, ILU(0) Pre-conditioner.

Appendix C. *Nested FJQNs: Numerical Results*

Table 20. Case Study 1: Throughput at Station 1.

Cfg	Sim	SCAE	Rel Err (%)	SCAI	Rel Err (%)	SCMI	Rel Err (%)
1	0.6023	0.6052	0.5	0.5317	-11.7	0.6053	0.5
2	0.7306	0.7308	0.0	0.6845	-6.3	0.7322	0.2
3	0.7894	0.7890	-0.1	0.7743	-1.9	0.7893	0.0
4	0.5653	0.6015	6.4	0.5113	-9.6	0.5934	5.0
5	0.7088	0.7296	2.9	0.6666	-6.0	0.7249	2.3
6	0.7844	0.7888	0.6	0.7676	-2.1	0.7892	0.6
7	0.5987	0.6046	1.0	0.5270	-12.0	0.6032	0.8
8	0.7288	0.7306	0.2	0.6811	-6.5	0.7312	0.3
9	0.7890	0.7890	0.0	0.7731	-2.0	0.7892	0.0
10	0.5859	0.5826	-0.6	0.5501	-6.1	0.5821	-0.6
11	0.7202	0.7190	-0.2	0.6948	-3.5	0.7171	-0.4
12	0.7870	0.7867	0.0	0.7770	-1.3	0.7855	-0.2
13	0.5377	0.5701	6.0	0.5353	-0.4	0.5600	4.1
14	0.6733	0.7093	5.3	0.6834	1.5	0.6891	2.3
15	0.7650	0.7843	2.5	0.7733	1.1	0.7719	0.9
16	0.5752	0.5793	0.7	0.5465	-5.0	0.5757	0.1
17	0.7112	0.7167	0.8	0.6925	-2.6	0.7113	0.0
18	0.7837	0.7862	0.3	0.7764	-0.9	0.7836	0.0

Table 21. Case Study 1: Queue Lengths at Station 1.

Cfg	Sim	SCAE	Rel Err (%)	SCAI	Rel Err (%)	SCMI	Rel Err (%)
1	1.1453	1.1763	2.7	0.9219	-19.5	1.1664	1.8
2	2.1761	2.2102	1.6	1.8073	-16.9	2.2162	1.8
3	3.4085	3.4482	1.2	3.0992	-9.1	3.4571	1.4
4	1.0896	1.1616	6.6	0.8671	-20.4	1.1324	3.9
5	2.1631	2.1945	1.5	1.6854	-22.1	2.1538	-0.4
6	3.6244	3.4415	-5.0	2.9682	-18.1	3.4515	-4.8
7	1.1468	1.1737	2.3	0.9090	-20.7	1.1596	1.1
8	2.1820	2.2077	1.2	1.7821	-18.3	2.2059	1.1
9	3.4396	3.4473	0.2	3.0755	-10.6	3.4515	0.3
10	1.0581	1.0934	3.3	0.9815	-7.2	1.0864	2.7
11	1.9758	2.0887	5.7	1.8914	-4.3	2.0676	4.6
12	3.1420	3.3754	7.4	3.1628	0.7	3.3455	6.5
13	1.0705	1.0544	-1.5	0.9386	-12.3	1.0391	-2.9
14	2.1438	2.0054	-6.5	1.8054	-15.8	1.9123	-10.8
15	3.8724	3.3091	-14.5	3.0826	-20.4	3.1304	-19.2
16	1.0683	1.0826	1.3	0.9705	-9.2	1.0709	0.2
17	2.0248	2.0682	2.1	1.8725	-7.5	2.0283	0.2
18	3.2876	3.3607	2.2	3.1476	-4.3	3.3028	0.5

Table 22. Case Study 1: Queue Lengths at Station 2.

Cfg	Sim	SCAE	Rel Err (%)	SCAI	Rel Err (%)	SCMI	Rel Err (%)
1	1.7522	1.7863	1.9	1.3492	-23.0	1.7638	0.7
2	4.3560	4.3930	0.8	3.3091	-24.0	4.3950	0.9
3	11.7101	11.7246	0.1	9.1420	-21.9	11.7752	0.6
4	1.5731	1.7606	11.9	1.2611	-19.8	1.7097	8.7
5	3.8840	4.3448	11.9	3.0170	-22.3	4.2343	9.0
6	10.6689	11.6510	9.2	8.3508	-21.7	11.7234	9.9
7	1.7319	1.7817	2.9	1.3281	-23.3	1.7527	1.2
8	4.3099	4.3851	1.7	3.2470	-24.7	4.3669	1.3
9	11.6013	11.7132	1.0	8.9895	-22.5	11.7234	1.1
10	1.6397	1.6422	0.2	1.4496	-11.6	1.6276	-0.7
11	4.0318	4.0469	0.4	3.5281	-12.5	3.9832	-1.2
12	10.9567	11.0577	0.9	9.5905	-12.5	10.8180	-1.3
13	1.4961	1.5773	5.4	1.3795	-7.8	1.5584	4.2
14	3.4802	3.8276	10.0	3.3131	-4.8	3.6447	4.7
15	9.0744	10.5348	16.1	9.0599	-0.2	9.6693	6.6
16	1.6010	1.6242	1.4	1.4315	-10.6	1.6040	0.2
17	3.8842	3.9916	2.8	3.4795	-10.4	3.8915	0.2
18	10.5381	10.9358	3.8	9.4836	-10.0	10.5476	0.1

Table 23. Case Study 1: Queue Lengths at Station 3.

Cfg	Sim	SCAE	Rel Err (%)	SCAI	Rel Err (%)	SCMI	Rel Err (%)
1	1.1596	1.1763	1.4	0.9219	-20.5	1.1664	0.6
2	2.2072	2.2102	0.1	1.8073	-18.1	2.2162	0.4
3	3.4450	3.4482	0.1	3.0992	-10.0	3.4571	0.4
4	1.0444	1.1616	11.2	0.8671	-17.0	1.1324	8.4
5	2.0106	2.1945	9.1	1.6854	-16.2	2.1538	7.1
6	3.3101	3.4415	4.0	2.9682	-10.3	3.4515	4.3
7	1.1465	1.1737	2.4	0.9090	-20.7	1.1596	1.1
8	2.1882	2.2077	0.9	1.7821	-18.6	2.2059	0.8
9	3.4441	3.4473	0.1	3.0755	-10.7	3.4515	0.2
10	1.0998	1.0934	-0.6	0.9815	-10.8	1.0864	-1.2
11	2.0972	2.0887	-0.4	1.8914	-9.8	2.0676	-1.4
12	3.3889	3.3754	-0.4	3.1628	-6.7	3.3455	-1.3
13	0.9861	1.0544	6.9	0.9386	-4.8	1.0391	5.4
14	1.8096	2.0054	10.8	1.8054	-0.2	1.9123	5.7
15	3.0025	3.3091	10.2	3.0826	2.7	3.1304	4.3
16	1.0680	1.0826	1.4	0.9705	-9.1	1.0709	0.3
17	2.0251	2.0682	2.1	1.8725	-7.5	2.0283	0.2
18	3.3067	3.3607	1.6	3.1476	-4.8	3.3028	-0.1

Table 24. Case Study 1: Queue Lengths at Station 4.

Cfg	Sim	SCAE	Rel Err (%)	SCAI	Rel Err (%)	SCMI	Rel Err (%)
1	0.1741	0.2328	33.7	0.3429	97.0	0.1755	0.8
2	0.2220	0.3231	45.5	0.4957	123.3	0.2230	0.5
3	0.2457	0.3708	50.9	0.6017	144.9	0.2457	0.0
4	0.1620	0.2267	39.9	0.3768	132.6	0.1716	5.9
5	0.2138	0.3187	49.1	0.5970	179.2	0.2203	3.0
6	0.2439	0.3690	51.3	0.7861	222.3	0.2453	0.6
7	0.1726	0.2309	33.8	0.3517	103.8	0.1748	1.3
8	0.2213	0.3213	45.2	0.5191	134.6	0.2226	0.6
9	0.2454	0.3696	50.6	0.6404	161.0	0.2456	0.1
10	0.3418	0.3301	-3.4	0.3495	2.3	0.3391	-0.8
11	0.4710	0.4482	-4.8	0.4859	3.2	0.4680	-0.6
12	0.5463	0.5135	-6.0	0.5740	5.1	0.5450	-0.2
13	0.3102	0.3651	17.7	0.4188	35.0	0.3249	4.7
14	0.4302	0.5441	26.5	0.6743	56.7	0.4437	3.1
15	0.5227	0.6657	27.4	0.8953	71.3	0.5305	1.5
16	0.3343	0.3397	1.6	0.3684	10.2	0.3348	0.1
17	0.4621	0.4718	2.1	0.5311	14.9	0.4626	0.1
18	0.5425	0.5481	1.0	0.6433	18.6	0.5427	0.0

Table 25. Case Study 1: Queue Lengths at Station 5.

Cfg	Sim	SCAE	Rel Err (%)	SCAI	Rel Err (%)	SCMI	Rel Err (%)
1	0.3815	0.4070	6.7	0.9510	149.3	0.3882	1.8
2	0.5145	0.5934	15.3	1.8678	263.0	0.5186	0.8
3	0.5858	0.7145	22.0	3.1908	444.7	0.5866	0.1
4	0.4674	0.4662	-0.3	1.0568	126.1	0.4870	4.2
5	0.7336	0.6719	-8.4	2.1610	194.6	0.7606	3.7
6	0.9236	0.7950	-13.9	3.8142	313.0	0.9255	0.2
7	0.4047	0.4190	3.5	0.9831	142.9	0.4117	1.7
8	0.5618	0.6072	8.1	1.9431	245.9	0.5665	0.8
9	0.6495	0.7271	11.9	3.3241	411.8	0.6497	0.0
10	0.8325	0.7964	-4.3	1.0197	22.5	0.8419	1.1
11	1.3964	1.2497	-10.5	1.9654	40.7	1.3987	0.2
12	1.9348	1.5792	-18.4	3.2634	68.7	1.9335	-0.1
13	1.0803	0.9004	-16.7	1.0896	0.9	1.0169	-5.9
14	2.0740	1.5203	-26.7	2.1357	3.0	2.0689	-0.2
15	3.4731	2.0443	-41.1	3.5996	3.6	3.5247	1.5
16	0.8982	0.8255	-8.1	1.0405	15.8	0.9006	0.3
17	1.5715	1.3182	-16.1	2.0058	27.6	1.5741	0.2
18	2.2921	1.6881	-26.4	3.3303	45.3	2.2912	0.0

Table 26. Case Study 1: Queue Lengths at Station 6.

Cfg	Sim	SCAE	Rel Err (%)	SCAI	Rel Err (%)	SCMI	Rel Err (%)
1	0.4077	0.4212	3.3	0.9919	143.3	0.4138	1.5
2	0.5630	0.6128	8.8	1.9494	246.3	0.5679	0.9
3	0.6489	0.7360	13.4	3.3039	409.2	0.6503	0.2
4	0.3749	0.4128	10.1	0.9693	158.5	0.4032	7.5
5	0.5360	0.5925	10.5	1.9577	265.2	0.5592	4.3
6	0.6416	0.7029	9.6	3.4707	440.9	0.6495	1.2
7	0.4036	0.4190	3.8	0.9831	143.6	0.4117	2.0
8	0.5603	0.6072	8.4	1.9431	246.8	0.5665	1.1
9	0.6496	0.7271	11.9	3.3241	411.7	0.6498	0.0
10	0.0615	0.3239	426.7	0.6541	963.6	0.0615	0.0
11	0.0771	0.6499	742.9	1.2093	1468.5	0.0771	0.0
12	0.0852	0.9162	975.4	1.9690	2211.0	0.0852	0.0
13	0.0566	0.3367	494.9	0.6321	1016.8	0.0591	4.4
14	0.0721	0.7171	894.6	1.1725	1526.2	0.0744	3.2
15	0.0825	1.0885	1219.4	1.9205	2227.9	0.0832	0.8
16	0.0604	0.3289	444.5	0.6502	976.5	0.0608	0.7
17	0.0764	0.6718	779.3	1.1983	1468.5	0.0764	0.0
18	0.0847	0.9641	1038.3	1.9452	2196.6	0.0850	0.4

Table 27. Case Study 1: Queue Lengths at Station 7.

Cfg	Sim	SCAE	Rel Err (%)	SCAI	Rel Err (%)	SCMI	Rel Err (%)
1	0.1679	0.2255	34.3	0.3390	101.9	0.1697	1.1
2	0.2120	0.3106	46.5	0.4854	129.0	0.2131	0.5
3	0.2336	0.3555	52.2	0.5852	150.5	0.2336	0.0
4	0.4678	0.2560	-45.3	0.4025	-14.0	0.2069	-55.8
5	0.7322	0.3685	-49.7	0.6717	-8.3	0.2946	-59.8
6	0.9220	0.4308	-53.3	0.9360	1.5	0.2799	-69.6
7	0.1726	0.2309	33.8	0.3517	103.8	0.1748	1.3
8	0.2211	0.3213	45.3	0.5191	134.8	0.2226	0.7
9	0.2453	0.3696	50.7	0.6404	161.1	0.2456	0.1
10	0.0298	0.1842	518.1	0.2467	727.9	0.0297	-0.3
11	0.0369	0.3037	723.0	0.3573	868.3	0.0368	-0.3
12	0.0405	0.3735	822.2	0.4433	994.6	0.0405	0.0
13	0.0594	0.2083	250.7	0.2902	388.6	0.0302	-49.2
14	0.0774	0.3929	407.6	0.4798	519.9	0.0384	-50.4
15	0.0897	0.5331	494.3	0.6708	647.8	0.0430	-52.1
16	0.0294	0.1901	546.6	0.2544	765.3	0.0296	0.7
17	0.0368	0.3228	777.2	0.3776	926.1	0.0368	0.0
18	0.0407	0.4041	892.9	0.4771	1072.2	0.0408	0.2

Table 28. Case Study 2: Throughput at Station 1.

Cfg	Sim	+/-	SCMI	Abs Error	Rel Err (%)
1	0.6093	0.0003	0.6119	0.00	0.4
2	0.7337	0.0002	0.7349	0.00	0.2
3	0.7896	0.0005	0.7898	0.00	0.0
4	0.6089	0.0001	0.5991	-0.01	-1.6
5	0.7328	0.0001	0.7323	0.00	-0.1
6	0.7891	0.0005	0.7892	0.00	0.0
7	0.6053	0.0001	0.6097	0.00	0.7
8	0.7324	0.0002	0.7339	0.00	0.2
9	0.7893	0.0005	0.7897	0.00	0.1
10	0.5981	0.0003	0.5893	-0.01	-1.5
11	0.7296	0.0003	0.7294	0.00	0.0
12	0.7891	0.0005	0.7889	0.00	0.0
13	0.5815	0.0003	0.5872	0.01	1.0
14	0.7167	0.0003	0.7222	0.01	0.8
15	0.7858	0.0003	0.7888	0.00	0.4
16	0.5960	0.0000	0.5970	0.00	0.2
17	0.7283	0.0003	0.7287	0.00	0.1
18	0.7885	0.0005	0.7889	0.00	0.1

Table 29. Case Study 2: Queue Lengths at Station 1.

Cfg	Sim	+/-	SCMI	Abs Error	Rel Err (%)
1	1.1747	0.0015	1.1919	0.02	1.5
2	2.2114	0.0055	2.2481	0.04	1.7
3	3.4281	0.0110	3.4734	0.05	1.3
4	1.2117	0.0017	1.1566	-0.06	-4.5
5	2.2916	0.0040	2.2271	-0.06	-2.8
6	3.5613	0.0059	3.4515	-0.11	-3.1
7	1.1725	0.0020	1.1851	0.01	1.1
8	2.2143	0.0015	2.2386	0.02	1.1
9	3.4398	0.0106	3.4682	0.03	0.8
10	1.1377	0.0020	1.1423	0.00	0.4
11	2.1783	0.0036	2.1868	0.01	0.4
12	3.4373	0.0155	3.4428	0.01	0.2
13	1.1259	0.0013	1.1158	-0.01	-0.9
14	2.2026	0.0037	2.1361	-0.07	-3.0
15	3.6384	0.0127	3.4380	-0.20	-5.5
16	1.1371	0.0013	1.1385	0.00	0.1
17	2.1699	0.0051	2.1799	0.01	0.5
18	3.4313	0.0160	3.4392	0.01	0.2

Table 30. Case Study 2: Queue Lengths at Station 2.

Cfg	Sim	+/-	SCMI	Abs Error	Rel Err (%)
1	1.7986	0.0025	1.8084	0.01	0.5
2	4.4667	0.0072	4.4904	0.02	0.5
3	11.8978	0.0201	11.9422	0.04	0.4
4	1.8259	0.0023	1.7522	-0.07	-4.0
5	4.4831	0.0050	4.4402	-0.04	-1.0
6	11.8849	0.0122	11.7232	-0.16	-1.4
7	1.7816	0.0020	1.7974	0.02	0.9
8	4.4201	0.0031	4.4647	0.04	1.0
9	11.8064	0.0198	11.8918	0.09	0.7
10	1.7267	0.0022	1.7231	0.00	-0.2
11	4.3111	0.0023	4.3116	0.00	0.0
12	11.6304	0.0207	11.6375	0.01	0.1
13	1.6674	0.0024	1.6823	0.01	0.9
14	4.1059	0.0045	4.1891	0.08	2.0
15	11.0907	0.0227	11.5945	0.50	4.5
16	1.7112	0.0019	1.7170	0.01	0.3
17	4.2848	0.0051	4.2931	0.01	0.2
18	11.5774	0.0210	11.6011	0.02	0.2

Table 31. Case Study 2: Queue Lengths at Station 3.

Cfg	Sim	+/-	SCMI	Abs Error	Rel Err (%)
1	1.1866	0.0011	1.1919	0.01	0.4
2	2.2390	0.0028	2.2481	0.01	0.4
3	3.4652	0.0156	3.4734	0.01	0.2
4	1.1958	0.0021	1.1566	-0.04	-3.3
5	2.2357	0.0031	2.2271	-0.01	-0.4
6	3.4491	0.0081	3.4515	0.00	0.1
7	1.1726	0.0015	1.1851	0.01	1.1
8	2.2191	0.0021	2.2386	0.02	0.9
9	3.4641	0.0113	3.4682	0.00	0.1
10	1.1428	0.0014	1.1423	0.00	0.0
11	2.1885	0.0034	2.1868	0.00	-0.1
12	3.4356	0.0083	3.4428	0.01	0.2
13	1.1017	0.0015	1.1158	0.01	1.3
14	2.0907	0.0046	2.1361	0.05	2.2
15	3.3511	0.0105	3.4380	0.09	2.6
16	1.1371	0.0013	1.1385	0.00	0.1
17	2.1762	0.0036	2.1799	0.00	0.2
18	3.4334	0.0082	3.4392	0.01	0.2

Table 32. Case Study 2: Queue Lengths at Station 4.

Cfg	Sim	+/-	SCMI	Abs Error	Rel Err (%)
1	0.0818	0.0002	0.0824	0.00	0.7
2	0.1007	0.0003	0.1009	0.00	0.2
3	0.1094	0.0001	0.1096	0.00	0.2
4	0.0817	0.0001	0.0805	0.00	-1.5
5	0.1006	0.0001	0.1010	0.00	0.4
6	0.1093	0.0001	0.1428	0.03	30.6
7	0.0813	0.0001	0.0820	0.00	0.9
8	0.1005	0.0002	0.1009	0.00	0.4
9	0.1096	0.0002	0.1095	0.00	-0.1
10	0.3073	0.0004	0.3078	0.00	0.2
11	0.4138	0.0008	0.4137	0.00	0.0
12	0.4683	0.0007	0.4686	0.00	0.1
13	0.2976	0.0003	0.3014	0.00	1.3
14	0.4035	0.0007	0.4083	0.00	1.2
15	0.4655	0.0007	0.4675	0.00	0.4
16	0.3060	0.0003	0.3070	0.00	0.3
17	0.4129	0.0005	0.4132	0.00	0.1
18	0.4683	0.0007	0.4684	0.00	0.0

Table 33. Case Study 2: Queue Lengths at Station 5.

Cfg	Sim	+/-	SCMI	Abs Error	Rel Err (%)
1	0.0805	0.0002	0.0809	0.00	0.5
2	0.0984	0.0002	0.0987	0.00	0.3
3	0.1067	0.0001	0.1068	0.00	0.1
4	0.0930	0.0004	0.0912	0.00	-1.9
5	0.1183	0.0005	0.1181	0.00	-0.2
6	0.1302	0.0003	0.1785	0.05	37.1
7	0.0811	0.0001	0.0820	0.00	1.1
8	0.1005	0.0003	0.1009	0.00	0.4
9	0.1094	0.0002	0.1095	0.00	0.1
10	0.2911	0.0003	0.2923	0.00	0.4
11	0.3842	0.0003	0.3847	0.00	0.1
12	0.4317	0.0005	0.4314	0.00	-0.1
13	0.4009	0.0010	0.3953	-0.01	-1.4
14	0.6165	0.0015	0.6109	-0.01	-0.9
15	0.7570	0.0032	0.7597	0.00	0.4
16	0.3058	0.0005	0.3070	0.00	0.4
17	0.4125	0.0006	0.4132	0.00	0.2
18	0.4677	0.0008	0.4684	0.00	0.1

Table 34. Case Study 2: Queue Lengths at Station 6.

Cfg	Sim	+/-	SCMI	Abs Error	Rel Err (%)
1	0.0817	0.0001	0.0824	0.00	0.9
2	0.1008	0.0002	0.1009	0.00	0.1
3	0.1093	0.0001	0.1095	0.00	0.2
4	0.0816	0.0001	0.0805	0.00	-1.3
5	0.1006	0.0001	0.1012	0.00	0.6
6	0.1093	0.0001	0.1429	0.03	30.7
7	0.0811	0.0002	0.0820	0.00	1.1
8	0.1005	0.0001	0.1008	0.00	0.3
9	0.1093	0.0001	0.1095	0.00	0.2
10	0.0274	0.0000	0.0276	0.00	0.7
11	0.0337	0.0000	0.0339	0.00	0.6
12	0.0367	0.0001	0.0368	0.00	0.3
13	0.0267	0.0001	0.0271	0.00	1.5
14	0.0332	0.0002	0.0336	0.00	1.2
15	0.0366	0.0001	0.0367	0.00	0.3
16	0.0273	0.0001	0.0276	0.00	1.1
17	0.0337	0.0001	0.0339	0.00	0.6
18	0.0367	0.0001	0.0368	0.00	0.3

Table 35. Case Study 2: Queue Lengths at Station 7.

Cfg	Sim	+/-	SCMI	Abs Error	Rel Err (%)
1	0.0804	0.0002	0.0809	0.00	0.6
2	0.0984	0.0002	0.0987	0.00	0.3
3	0.1068	0.0001	0.1068	0.00	0.0
4	0.0927	0.0006	0.0912	0.00	-1.6
5	0.1183	0.0000	0.1255	0.01	6.1
6	0.1309	0.0004	0.1785	0.05	36.4
7	0.0811	0.0001	0.0820	0.00	1.1
8	0.1003	0.0002	0.1008	0.00	0.5
9	0.1092	0.0002	0.1095	0.00	0.3
10	0.0274	0.0000	0.0274	0.00	0.0
11	0.0336	0.0000	0.0336	0.00	0.0
12	0.0364	0.0001	0.0365	0.00	0.3
13	0.0280	0.0002	0.0284	0.00	1.4
14	0.0351	0.0003	0.0357	0.00	1.7
15	0.0390	0.0002	0.0394	0.00	1.0
16	0.0273	0.0001	0.0276	0.00	1.1
17	0.0338	0.0001	0.0339	0.00	0.3
18	0.0367	0.0001	0.0368	0.00	0.3

Table 36. Case Study 2: Queue Lengths at Station 8.

Cfg	Sim	+/-	SCMI	Abs Error	Rel Err (%)
1	0.1763	0.0002	0.1777	0.00	0.8
2	0.2233	0.0001	0.2240	0.00	0.3
3	0.2453	0.0003	0.2459	0.00	0.2
4	0.1763	0.0003	0.1736	0.00	-1.5
5	0.2229	0.0004	0.2216	0.00	-0.6
6	0.2453	0.0002	0.3333	0.09	35.9
7	0.1747	0.0002	0.1770	0.00	1.3
8	0.2226	0.0001	0.2237	0.00	0.5
9	0.2452	0.0005	0.2457	0.00	0.2
10	0.0307	0.0001	0.0308	0.00	0.3
11	0.0378	0.0001	0.0378	0.00	0.0
12	0.0411	0.0001	0.0411	0.00	0.0
13	0.0298	0.0001	0.0302	0.00	1.3
14	0.0372	0.0001	0.0374	0.00	0.5
15	0.0409	0.0001	0.0411	0.00	0.5
16	0.0307	0.0001	0.0307	0.00	0.0
17	0.0378	0.0001	0.0378	0.00	0.0
18	0.0411	0.0001	0.0411	0.00	0.0

Table 37. Case Study 2: Queue Lengths at Station 9.

Cfg	Sim	+/-	SCMI	Abs Error	Rel Err (%)
1	0.3869	0.0005	0.3932	0.01	1.6
2	0.5182	0.0007	0.5213	0.00	0.6
3	0.5869	0.0007	0.5871	0.00	0.0
4	0.2198	0.0006	0.5079	0.29	131.1
5	0.3007	0.0012	0.7696	0.47	155.9
6	0.3413	0.0008	1.0000	0.66	193.0
7	0.4103	0.0009	0.4176	0.01	1.8
8	0.5667	0.0011	0.5699	0.00	0.6
9	0.6487	0.0016	0.6499	0.00	0.2
10	0.0624	0.0002	0.0625	0.00	0.2
11	0.0773	0.0002	0.0772	0.00	-0.1
12	0.0840	0.0003	0.0839	0.00	-0.1
13	0.0314	0.0002	0.0659	0.03	109.9
14	0.0398	0.0002	0.0839	0.04	110.8
15	0.0441	0.0003	0.0932	0.05	111.3
16	0.0630	0.0002	0.0632	0.00	0.3
17	0.0784	0.0002	0.0785	0.00	0.1
18	0.0855	0.0003	0.0859	0.00	0.5

Table 38. Case Study 2: Queue Lengths at Station 10.

Cfg	Sim	+/-	SCMI	Abs Error	Rel Err (%)
1	0.1751	0.0003	0.1778	0.00	1.5
2	0.2214	0.0003	0.2241	0.00	1.2
3	0.2435	0.0005	0.2459	0.00	1.0
4	0.1795	0.0003	0.1736	-0.01	-3.3
5	0.2285	0.0004	0.2230	-0.01	-2.4
6	0.2526	0.0003	0.3333	0.08	31.9
7	0.1746	0.0002	0.1771	0.00	1.4
8	0.2227	0.0003	0.2238	0.00	0.5
9	0.2455	0.0004	0.2474	0.00	0.8
10	0.3440	0.0004	0.3517	0.01	2.2
11	0.4668	0.0007	0.4806	0.01	3.0
12	0.5304	0.0006	0.5489	0.02	3.5
13	0.3585	0.0005	0.3444	-0.01	-3.9
14	0.5251	0.0005	0.4740	-0.05	-9.7
15	0.6382	0.0012	0.5473	-0.09	-14.2
16	0.3500	0.0002	0.3508	0.00	0.2
17	0.4793	0.0006	0.4799	0.00	0.1
18	0.5487	0.0011	0.5505	0.00	0.3

Appendix D. *The Analytical Airfield Model: Numerical Results*

Table 39. Effect of Mean Interarrival Time on Airfield Throughput.

Arr Rate	Ground Delay			No Ground Delay		
	Sim	AAM	Rel Err	Sim	AAM	Rel Err
0.30	2.2340	2.2110	-1.0	2.6296	2.5702	-2.3
0.40	1.9596	1.9665	0.4	2.2288	2.1835	-2.0
0.50	1.7158	1.7321	0.9	1.8844	1.8757	-0.5
0.60	1.5096	1.5255	1.1	1.6139	1.6057	-0.5
0.70	1.3378	1.3511	1.0	1.4043	1.3977	-0.5
0.80	1.1952	1.2062	0.9	1.2385	1.2347	-0.3
0.90	1.0777	1.0862	0.8	1.1037	1.1034	0.0

Table 40. Effect of Mean Interarrival Time on Response Time.

Arr Rate	Ground Delay					No Ground Delay				
	Sim	AAM	Rel Err	Determ	Rel Err	Sim	AAM	Rel Err	Determ	Rel Err
0.30	2.7514	2.8683	4.2	2.6567	-3.4	2.0462	2.1223	3.7	1.5299	-27.9
0.40	2.7448	2.8407	3.5	2.6567	-3.2	1.9404	2.0735	6.9	1.5299	-26.2
0.50	2.7401	2.8205	2.9	2.6567	-3.0	1.8693	2.0171	7.9	1.5299	-24.2
0.60	2.7364	2.8094	2.7	2.6567	-2.9	1.8241	1.8877	3.5	1.5299	-19.0
0.70	2.7337	2.8011	2.5	2.6567	-2.8	1.7911	1.8291	2.1	1.5299	-16.4
0.80	2.7318	2.7933	2.3	2.6567	-2.8	1.7676	1.6270	-8.0	1.5299	-6.0
0.90	2.7301	2.7836	2.0	2.6567	-2.7	1.7502	1.6830	-3.8	1.5299	-9.1

Table 41. Effect of Mean Interarrival Time on Aircraft on Station.

Arr Rate	Ground Delay			No Ground Delay		
	Sim	AAM	Rel Err	Sim	AAM	Rel Err
0.30	6.1466	6.3418	3.2	5.3806	5.3374	-0.8
0.40	5.3789	5.5863	3.9	4.3251	4.3775	1.2
0.50	4.7016	4.8855	3.9	3.5227	3.6212	2.8
0.60	4.1314	4.2858	3.7	2.9433	2.9446	0.0
0.70	3.6574	3.7847	3.5	2.5153	2.5032	-0.5
0.80	3.2652	3.3693	3.2	2.1895	2.0013	-8.6
0.90	2.9420	3.0235	2.8	1.9312	1.8389	-4.8

Table 42. Effect of Constrained Resources on Throughput, Analytical Results.

No Refuel Servers	No Cargo Servers, Ground Delay					No Cargo Servers, No Ground Delay				
	1	2	3	4	5	1	2	3	4	5
1	0.9379	1.0822	1.0835	1.0838	1.0840	0.9402	1.0686	1.0744	1.0754	1.0756
2	0.9379	1.0847	1.0860	1.0864	1.0866	0.9398	1.0809	1.0824	1.0830	1.0830
3	0.9379	1.0849	1.0861	1.0865	1.0867	0.9397	1.0809	1.0824	1.0830	1.0831
4	0.9379	1.0849	1.0862	1.0865	1.0868	0.9397	1.0809	1.0824	1.0830	1.0831
5	0.9379	1.0849	1.0862	1.0865	1.0868	0.9397	1.0809	1.0824	1.0830	1.0831
6	0.9379	1.0849	1.0862	1.0865	1.0868	0.9397	1.0809	1.0824	1.0830	1.0831
7	0.9379	1.0849	1.0862	1.0865	1.0868	0.9397	1.0809	1.0824	1.0830	1.0831
8	0.9379	1.0849	1.0862	1.0865	1.0868	0.9397	1.0809	1.0824	1.0830	1.0831

Table 43. Effect of Constrained Resources on Response Time, Analytical Results.

No Refuel Servers	No Cargo Servers, Ground Delay					No Cargo Servers, No Ground Delay				
	1	2	3	4	5	1	2	3	4	5
1	5.0244	2.8619	2.8485	2.8446	2.8420	4.7988	2.2030	2.1183	2.1058	2.1040
2	5.0189	2.8034	2.7895	2.7854	2.7828	4.7957	1.9056	1.8747	1.8639	1.8639
3	5.0185	2.7985	2.7846	2.7805	2.7778	4.7956	1.9028	1.8720	1.8611	1.8594
4	5.0184	2.7977	2.7838	2.7797	2.7770	4.7957	1.9026	1.8718	1.8609	1.8592
5	5.0185	2.7975	2.7836	2.7795	2.7769	4.7957	1.9026	1.8717	1.8609	1.8592
6	5.0184	2.7975	2.7836	2.7795	2.7768	4.7957	1.9026	1.8717	1.8609	1.8592
7	5.0184	2.7975	2.7836	2.7795	2.7768	4.7957	1.9026	1.8718	1.8609	1.8592
8	5.0184	2.7975	2.7836	2.7795	2.7768	4.7957	1.9026	1.8718	1.8609	1.8592

Table 44. Effect of Constrained Resources on Aircraft on Station, Analytical Results.

No Refuel Servers	No Cargo Servers, Ground Delay					No Cargo Servers, No Ground Delay				
	1	2	3	4	5	1	2	3	4	5
1	4.7121	3.0973	3.0862	3.0830	3.0808	4.5117	2.3541	2.2758	2.2645	2.2631
2	4.7073	3.0409	3.0294	3.0259	3.0237	4.5069	2.0597	2.0292	2.0186	2.0186
3	4.7069	3.0360	3.0245	3.0210	3.0188	4.5067	2.0568	2.0262	2.0157	2.0140
4	4.7068	3.0352	3.0237	3.0202	3.0179	4.5066	2.0565	2.0260	2.0154	2.0138
5	4.7069	3.0350	3.0235	3.0201	3.0178	4.5066	2.0565	2.0260	2.0154	2.0138
6	4.7068	3.0350	3.0235	3.0201	3.0178	4.5066	2.0565	2.0260	2.0154	2.0138
7	4.7068	3.0350	3.0235	3.0201	3.0178	4.5066	2.0565	2.0260	2.0154	2.0138
8	4.7068	3.0350	3.0235	3.0201	3.0178	4.5066	2.0565	2.0260	2.0154	2.0138

Table 45. Effect of Constrained Resources on Throughput, Simulation Results.

No Refuel Servers	No Cargo Servers, Ground Delay					No Cargo Servers, No Ground Delay				
	1	2	3	4	5	1	2	3	4	5
1	0.9347	1.0652	1.0649	1.0663	1.0660	0.9362	1.0817	1.0911	1.0932	1.0935
2	0.9355	1.0728	1.0743	1.0758	1.0752	0.9374	1.0949	1.1011	1.1058	1.1032
3	0.9355	1.0709	1.0766	1.0735	1.0767	0.9379	1.0942	1.1031	1.1055	1.1068
4	0.9347	1.0717	1.0764	1.0766	1.0733	0.9364	1.0939	1.1027	1.1047	1.1061
5	0.9340	1.0727	1.0768	1.0770	1.0776	0.9361	1.0946	1.1029	1.1054	1.1042
6	0.9359	1.0737	1.0777	1.0756	1.0759	0.9375	1.0932	1.1037	1.1052	1.1068
7	0.9346	1.0728	1.0786	1.0752	1.0781	0.9366	1.0932	1.1048	1.1056	1.1069
8	0.9346	1.0728	1.0786	1.0752	1.0781	0.9366	1.0932	1.1043	1.1042	1.1068

Table 46. Effect of Constrained Resources on Response Time, Simulation Results.

No Refuel Servers	No Cargo Servers, Ground Delay					No Cargo Servers, No Ground Delay				
	1	2	3	4	5	1	2	3	4	5
1	5.0329	3.0251	2.9696	2.9662	2.9677	4.8692	2.3632	2.1643	2.1304	2.1246
2	5.0095	2.8188	2.7417	2.7364	2.7353	4.8478	2.0632	1.7861	1.7385	1.7303
3	5.0009	2.8102	2.7307	2.7246	2.7244	4.8234	2.0407	1.7565	1.7058	1.6946
4	4.9987	2.8081	2.7301	2.7239	2.7230	4.8516	2.0397	1.7252	1.7011	1.6905
5	5.0005	2.8083	2.7301	2.7239	2.7233	4.8316	2.0440	1.7514	1.7016	1.6899
6	5.0142	2.8086	2.7301	2.7239	2.7230	4.8296	2.0443	1.7502	1.7005	1.6899
7	5.0010	2.8093	2.7302	2.7241	2.7231	4.8307	2.0442	1.7516	1.7006	1.6902
8	5.0010	2.8093	2.7302	2.7241	2.7231	4.8307	2.0442	1.7518	1.7012	1.6898

Table 47. Effect of Constrained Resources on Aircraft on Station, Simulation Results.

No Refuel Servers	No Cargo Servers, Ground Delay					No Cargo Servers, No Ground Delay				
	1	2	3	4	5	1	2	3	4	5
1	4.7042	3.2224	3.1623	3.1631	3.1639	4.5585	2.5563	2.3616	2.3290	2.3235
2	4.6867	3.0234	2.9450	2.9435	2.9406	4.5443	2.2591	1.9660	1.9221	1.9086
3	4.6783	3.0088	2.9395	2.9245	2.9330	4.5241	2.2331	1.9370	1.8853	1.8753
4	4.6725	3.0089	2.9385	2.9322	2.9225	4.5431	2.2313	1.9321	1.8788	1.8696
5	4.6706	3.0119	2.9395	2.9333	2.9344	4.5228	2.2376	1.9311	1.8806	1.8657
6	4.6926	3.0152	2.9420	2.9297	2.9293	4.5278	2.2350	1.9312	1.8791	1.8687
7	4.6738	3.0132	2.9444	2.9288	2.9355	4.5244	2.2349	1.9347	1.8798	1.8706
8	4.6738	3.0132	2.9444	2.9288	2.9355	4.5244	2.2349	1.9341	1.8783	1.8701

Table 48. Effect of Constrained Resources on Throughput, Relative Errors (Percent).

No Refuel Servers	No Cargo Servers, Ground Delay					No Cargo Servers, No Ground Delay				
	1	2	3	4	5	1	2	3	4	5
1	0.3	1.6	1.7	1.6	1.7	0.4	-1.2	-1.5	-1.6	-1.6
2	0.3	1.1	1.1	1.0	1.1	0.3	-1.3	-1.7	-2.1	-1.8
3	0.3	1.3	0.9	1.2	0.9	0.2	-1.2	-1.9	-2.0	-2.1
4	0.3	1.2	0.9	0.9	1.3	0.4	-1.2	-1.8	-2.0	-2.1
5	0.4	1.1	0.9	0.9	0.9	0.4	-1.3	-1.9	-2.0	-1.9
6	0.2	1.0	0.8	1.0	1.0	0.2	-1.1	-1.9	-2.0	-2.1
7	0.4	1.1	0.7	1.1	0.8	0.3	-1.1	-2.0	-2.0	-2.2
8	0.4	1.1	0.7	1.1	0.8	0.3	-1.1	-2.0	-1.9	-2.1

Table 49. Effect of Constrained Resources on Response Time, Relative Errors (Percent).

No Refuel Servers	No Cargo Servers, Ground Delay					No Cargo Servers, No Ground Delay				
	1	2	3	4	5	1	2	3	4	5
1	-0.2	-5.4	-4.1	-4.1	-4.2	-1.4	-6.8	-2.1	-1.2	-1.0
2	0.2	-0.5	1.7	1.8	1.7	-1.1	-7.6	5.0	7.2	7.7
3	0.4	-0.4	2.0	2.1	2.0	-0.6	-6.8	6.6	9.1	9.7
4	0.4	-0.4	2.0	2.0	2.0	-1.2	-6.7	8.5	9.4	10.0
5	0.4	-0.4	2.0	2.0	2.0	-0.7	-6.9	6.9	9.4	10.0
6	0.1	-0.4	2.0	2.0	2.0	-0.7	-6.9	6.9	9.4	10.0
7	0.3	-0.4	2.0	2.0	2.0	-0.7	-6.9	6.9	9.4	10.0
8	0.3	-0.4	2.0	2.0	2.0	-0.7	-6.9	6.9	9.4	10.0

Table 50. Effect of Constrained Resources on Aircraft on Station, Relative Errors (Percent).

No Refuel Servers	No Cargo Servers, Ground Delay					No Cargo Servers, No Ground Delay				
	1	2	3	4	5	1	2	3	4	5
1	0.2	-3.9	-2.4	-2.5	-2.6	-1.0	-7.9	-3.6	-2.8	-2.6
2	0.4	0.6	2.9	2.8	2.8	-0.8	-8.8	3.2	5.0	5.8
3	0.6	0.9	2.9	3.3	2.9	-0.4	-7.9	4.6	6.9	7.4
4	0.7	0.9	2.9	3.0	3.3	-0.8	-7.8	4.9	7.3	7.7
5	0.8	0.8	2.9	3.0	2.8	-0.4	-8.1	4.9	7.2	7.9
6	0.3	0.7	2.8	3.1	3.0	-0.5	-8.0	4.9	7.3	7.8
7	0.7	0.7	2.7	3.1	2.8	-0.4	-8.0	4.7	7.2	7.7
8	0.7	0.7	2.7	3.1	2.8	-0.4	-8.0	4.8	7.3	7.7

Table 51. Effect of Constrained Resources on Response Time, Deterministic Results.

No Refuel Servers	No Cargo Servers, Ground Delay					No Cargo Servers, No Ground Delay				
	1	2	3	4	5	1	2	3	4	5
1	2.6567	2.6567	2.6567	2.6567	2.6567	1.5299	1.5299	1.5299	1.5299	1.5299
2	2.6567	2.6567	2.6567	2.6567	2.6567	1.5299	1.5299	1.5299	1.5299	1.5299
3	2.6567	2.6567	2.6567	2.6567	2.6567	1.5299	1.5299	1.5299	1.5299	1.5299
4	2.6567	2.6567	2.6567	2.6567	2.6567	1.5299	1.5299	1.5299	1.5299	1.5299
5	2.6567	2.6567	2.6567	2.6567	2.6567	1.5299	1.5299	1.5299	1.5299	1.5299
6	2.6567	2.6567	2.6567	2.6567	2.6567	1.5299	1.5299	1.5299	1.5299	1.5299
7	2.6567	2.6567	2.6567	2.6567	2.6567	1.5299	1.5299	1.5299	1.5299	1.5299
8	2.6567	2.6567	2.6567	2.6567	2.6567	1.5299	1.5299	1.5299	1.5299	1.5299

Table 52. Effect of Constrained Resources on Response Time, Relative Errors in Deterministic Results (Percent).

No Refuel Servers	No Cargo Servers, Ground Delay					No Cargo Servers, No Ground Delay				
	1	2	3	4	5	1	2	3	4	5
2	-47.2	-12.2	-10.5	-10.4	-10.5	-68.6	-35.3	-29.3	-28.2	-28.0
3	-47.0	-5.8	-3.1	-2.9	-2.9	-68.4	-25.8	-14.3	-12.0	-11.6
3	-46.9	-5.5	-2.7	-2.5	-2.5	-68.3	-25.0	-12.9	-10.3	-9.7
4	-46.9	-5.4	-2.7	-2.5	-2.4	-68.5	-25.0	-11.3	-10.1	-9.5
5	-46.9	-5.4	-2.7	-2.5	-2.4	-68.3	-25.2	-12.6	-10.1	-9.5
6	-47.0	-5.4	-2.7	-2.5	-2.4	-68.3	-25.2	-12.6	-10.0	-9.5
7	-46.9	-5.4	-2.7	-2.5	-2.4	-68.3	-25.2	-12.7	-10.0	-9.5
8	-46.9	-5.4	-2.7	-2.5	-2.4	-68.3	-25.2	-12.7	-10.1	-9.5

Table 53. Effect of Squared Coefficient of Variation on Airfield Throughput.

Case	CV^2	Ground Delay			No Ground Delay		
		Sim	AAM	Rel Err	Sim	AAM	Rel Err
Baseline	0.50	1.0777	1.0862	0.8	1.1037	1.1034	0.0
	0.75	1.0742	1.0854	1.0	1.1038	1.1010	-0.3
	1.00	1.0747	1.0892	1.3	1.1023	1.1057	0.3
Constrained	0.50	1.0728	1.0847	1.1	1.0949	1.0809	-1.3
	0.75	1.0683	1.0829	1.4	1.0875	1.0954	0.7
	1.00	1.0661	1.0810	1.4	1.0875	1.0909	0.3

Table 54. Effect of Squared Coefficient of Variation on Time on Response Time.

Case	CV^2	Ground Delay					No Ground Delay				
		Sim	AAM	Rel Err	Determ	Rel Err	Sim	AAM	Rel Err	Determ	Rel Err
Baseline	0.50	2.7301	2.7836	2.0	2.6567	-2.7	1.7502	1.6830	-3.8	1.5299	-12.6
	0.75	2.8120	2.7901	-0.8	2.6567	-5.5	1.8457	1.8476	0.1	1.5299	-17.1
	1.00	2.8396	2.7670	-2.6	2.6567	-6.4	1.8720	1.6522	-11.7	1.5299	-18.3
Constrained	0.50	2.8188	2.8034	-0.5	2.6567	-5.8	2.0632	1.9056	-7.6	1.5299	-25.8
	0.75	2.9326	2.8213	-3.8	2.6567	-9.4	2.1780	2.0405	-6.3	1.5299	-29.8
	1.00	2.9696	2.9037	-2.2	2.6567	-10.5	2.2104	2.0188	-8.7	1.5299	-30.8

Table 55. Effect of Squared Coefficient of Variation on Aircraft on Station.

Case	CV^2	Ground Delay			No Ground Delay		
		Sim	AAM	Rel Err	Sim	AAM	Rel Err
Baseline	0.50	2.9420	3.0235	2.8	1.9312	1.8389	-4.8
	0.75	3.0202	3.0283	0.3	2.0381	1.9919	-2.3
	1.00	3.0520	3.0138	-1.3	2.0641	1.8287	-11.4
Constrained	0.50	3.0234	3.0409	0.6	2.2591	2.0597	-8.8
	0.75	3.1329	3.0553	-2.5	2.3685	2.1783	-8.0
	1.00	3.1660	3.1391	-0.8	2.4039	2.2022	-8.4

Bibliography

- [1] Almeida, V. A. F. and L. W. Dowdy. "Performance analysis of a scheme for concurrency/synchronization using queueing network models," *International Journal of Parallel Programming*, 15(6):529-550 (1986).
- [2] Ammar, M. H. and S. B. Gershwin. "Equivalence relations in queueing models of fork/join networks with blocking," *Performance Evaluation*, 10(3):233-245 (1989).
- [3] Arnoldi, W. E. "The principle of minimized iteration in the solution of the matrix eigenvalue problem," *Quarterly Journal of Applied Mathematics*, 9:17-29 (1951).
- [4] Avi-Itzhak, B. and S. Halfin. "Non-preemptive priorities in simple fork-join queues." *Queueing, Performance and Control in ATM, ITC-13 Workshops: Proceedings of the 13th International Teletraffic Conference*, edited by J. W. Cohen and C. D. Pack, 231-238. Amsterdam: Elsevier Science Publishers B. V. (North-Holland), 1991.
- [5] Avi-Itzhak, B. and D. Heyman. "Approximate queueing models for multiprogramming computer systems," *Operations Research*, 21(6):1212-1230 (1973).
- [6] Axelsson, O. *Iterative Solution Methods*. Cambridge: Cambridge University Press, 1994.
- [7] Axelsson, O. and V. A. Barker. *Finite Element Solution of Boundary Value Problems: Theory and Computation*. Computer Science and Applied Mathematics, New York: Academic Press, 1984.
- [8] Baccelli, F. *Two parallel queues created by arrivals with two demands: the M/G/2 symmetrical case*. Technical Report 426. Le Chesnay, France: Institut National de Recherche en Informatique et en Automatique, 1985.
- [9] Baccelli, F. and Z. Liu. "On the execution of parallel programs on multiprocessor systems—a queuing theory approach," *Journal of the Association for Computing Machinery*, 37(2):373-414 (1990).
- [10] Baccelli, F. and A. M. Makowski. "Simple computable bounds for the fork-join queue." *Proceedings of the 19th Annual Conference on Information Sciences and Systems*. 436-441. Baltimore: Johns Hopkins University, 1985.
- [11] Baccelli, F. and A. M. Makowski. "Multidimensional stochastic ordering and associated random variables," *Operations Research*, 37(3):478-487 (1989).

- [12] Baccelli, F. and A. M. Makowski. "Queueing models for systems with synchronization constraints," *Proceedings of the IEEE*, 77(1):138-161 (1989).
- [13] Baccelli, F. and A. M. Makowski. "Synchronization in queueing systems." *Stochastic Analysis of Computer and Communication Systems*, edited by H. Takagi, 57-129. Amsterdam: Elsevier Science Publishers B. V. (North-Holland), 1990. Reprint of Reference [12].
- [14] Baccelli, F., and others. "The fork-join queue and related systems with synchronization constraints: stochastic ordering and computable bounds," *Advances in Applied Probability*, 21:629-660 (1989).
- [15] Baccelli, F. and W. A. Massey. *Series-Parallel Fork-Join Queueing Networks and Their Stochastic Ordering*. Technical Report 534. Le Chesnay, France: Institut National de Recherche en Informatique et en Automatique, 1986.
- [16] Baccelli, F., and others. "Acyclic fork-join queueing networks," *Journal of the Association for Computing Machinery*, 36(3):615-642 (1989).
- [17] Balsamo, S. and L. Donatiello. "Approximate performance analysis of parallel processing systems." *Decentralized Systems: Proceedings of the IFIP WG 10.3 Working Conference on Decentralized Systems, Lyons, France, 11-13 December 1989*, edited by M. Cosnard and C. Girault, 325-336. Amsterdam: Elsevier Science Publishers B. V. (North-Holland), 1990.
- [18] Barrett, R., and others. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. Philadelphia: Society for Industrial and Applied Mathematics, 1994.
- [19] Baskett, F., and others. "Open, closed, and mixed networks of queues with different classes of customers," *Journal of the Association for Computing Machinery*, 22(2):248-260 (1975).
- [20] Baynat, B. and Y. Dallery. *A Product-Form Approximation Method for General Closed Queueing Networks with Several Classes of Customers*. Technical Report MASI 91.50. Paris: Laboratoire de Méthodologie et Architecture des Systèmes Informatiques, Institut Blaise Pascal, Université Pierre et Marie Curie, 1991.
- [21] Baynat, B. and Y. Dallery. *An Approximation Method for General Closed Queueing Networks with Fork/Join Mechanisms*. Technical Report. Paris: Laboratoire de Méthodologie et Architecture des Systèmes Informatiques, Institut Blaise Pascal, Université Pierre et Marie Curie, 1993.
- [22] Baynat, B. and Y. Dallery. "A decomposition approximation for closed queueing networks with fork/join subnetworks." *IFIP Transactions A (Computer*

- Science and Technology*), A-39, edited by M. Cosnard and R. Puigjaner, 199–210. Amsterdam: Elsevier Science Publishers B. V. (North-Holland), 1993.
- [23] Baynat, B. and Y. Dallery. "A unified view of product-form approximation techniques for general closed queueing networks," *Performance Evaluation*, 18(3):205–224 (1993).
 - [24] Baynat, B. and Y. Dallery. *Approximate analysis of multi-class synchronized closed queueing networks*. Technical Report. Paris: Laboratoire de Méthodologie et Architecture des Systèmes Informatiques, Institut Blaise Pascal, Université Pierre et Marie Curie, 1994.
 - [25] Baynat, B. and Y. Dallery. "Approximate analysis of multi-class synchronized closed queueing networks." *MASCOTS '95: Proceedings of the Third International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, edited by P. Dowd and E. Gelenbe, 23–27. Los Alamitos CA: IEEE Computer Society Press, 1995.
 - [26] Baynat, B., and others. "A decomposition approximation method for multiclass BCMP queueing networks with multiple server stations," *Annals of Operational Research*, 48(1–4):273–294 (1994).
 - [27] Bondi, A. B. and W. Whitt. "The influence of service time variability in a closed network of queues," *Performance Evaluation*, 6(3):219–234 (1986).
 - [28] Brun, M. A. and G. Fayolle. "The distribution of the transaction processing time in a simple fork-join system." *Computer Performance and Reliability: Proceedings of the 2nd International MCPWR Workshop*, edited by G. Iazolla and others, 203–212. Amsterdam: Elsevier Science Publishers B. V. (North-Holland), 1988.
 - [29] Buzen, J. P. "Computational algorithms for closed queueing networks with exponential servers," *Communications of the ACM*, 16(9):527–531 (1973).
 - [30] Campos, J., and others. "Properties and performance bounds for closed free choice synchronized monoclase queueing networks," *IEEE Transactions on Automatic Control*, 36(12):1368–1382 (1991).
 - [31] Chan, T. F. and T. Szeto. "Composite Step Methods for Solving Nonsymmetric Linear Systems," *SIAM Journal on Scientific Computing*, 17(6):1491–1508 (1996).
 - [32] Chandy, K. M., and others. "Approximate analysis of general queueing networks," *IBM Journal of Research and Development*, 19(1):43–49 (1975).
 - [33] Chandy, K. M., and others. "Parametric analysis of queueing networks," *IBM Journal of Research and Development*, 19(1):36–42 (1975).

- [34] Chatelin, F. *Spectral Approximation of Linear Operators*. New York: Academic Press, 1983.
- [35] Conway, A. E. and N. D. Georganas. *Queueing Networks—Exact Computational Algorithms*. Cambridge MA: The MIT Press, 1989.
- [36] Courtois, P. J. *Decomposability*. New York: Academic Press, 1977.
- [37] Cox, D. R. "A use of complex probabilities in the theory of stochastic processes," *Proceedings of the Cambridge Philosophical Society*, 51(2):313–319 (1955).
- [38] Dallery, Y. "Approximate analysis of general open queueing networks with restricted capacity," *Performance Evaluation*, 11(3):209–222 (1990).
- [39] Dallery, Y. and X. Cao. "Operational analysis of stochastic closed queueing networks," *Performance Evaluation*, 14(1):43–61 (1992).
- [40] Dallery, Y., and others. "Equivalence, reversibility, symmetry and concavity properties in fork-join queueing networks with blocking," *Journal of the Association for Computing Machinery*, 41(5):903–942 (1994).
- [41] DeKlein, S. J. *Two Parallel Queues With Simultaneous Service Demands*. Technical Report 360. Utrecht, The Netherlands: Department of Mathematics, University of Utrecht, 1984.
- [42] Di Mascolo, M., and others. *Queueing Network Modeling and Analysis of Generalized Kanban Systems*. Technical Report. Saint Martin d'Hères, France: Laboratoire de Automatique de Grenoble, 1993.
- [43] Di Mascolo, M., and others. "An analytical method for performance evaluation of kanban controlled production systems," *Operations Research*, 44(1):50–64 (1996).
- [44] Dietz, D. C. and R. C. Jenkins. "Analysis of aircraft sortie generation with the use of a fork-join queueing network model," *Naval Research Logistics*, 44(2):153–164 (1997).
- [45] Ding, Y. *On Performance Control of Real-Time Systems*. PhD dissertation, The University of Connecticut, 1991 (DA9215426).
- [46] Duda, A. and T. Czachórski. "Performance evaluation of fork and join synchronization primitives," *Acta Informatica*, 24(5):525–553 (1987).
- [47] Evans, M., and others. *Statistical Distributions* (Second Edition). New York: John Wiley and Sons, 1993.
- [48] Flatto, L. "Two parallel queues created by arrivals with two demands, II," *SIAM Journal of Applied Mathematics*, 45(5):861–878 (1985).

- [49] Flatto, L. and S. Hahn. "Two parallel queues created by arrivals with two demands, I," *SIAM Journal of Applied Mathematics*, 44(5):1041-1053 (1984).
- [50] Fletcher, R. *Conjugate Gradient Methods for Indefinite Systems*, 506. Lecture Notes in Mathematics, 73-89. Berlin: Springer-Verlag, 1976.
- [51] Freund, R. and N. Nachtigal. "QMR: a quasi-minimal residual method for non-Hermitian linear systems," *Numerical Mathematics*, 60:315-339 (1991).
- [52] Gelenbe, E. and G. Pujolle. *Introduction to Queueing Networks*. Chichester, UK: John Wiley and Sons, 1987. Translated from French by J. C. C. Nelson.
- [53] Gershwin, S. B. "Modeling and analysis of assembly/disassembly networks." *Proceedings of the 1986 IEEE Conference on Systems, Man and Cybernetics*. 687-691. New York: IEEE Press, 1986.
- [54] Gershwin, S. B. "Assembly/disassembly systems: an efficient decomposition algorithm for tree-structured networks," *IEEE Transactions*, 23(4):302-314 (1991).
- [55] Grassman, W. K., and others. "Regenerative analysis and steady state distributions for Markov chains," *Operations Research*, 33(5):1107-1116 (1985).
- [56] Gün, L. and A. M. Makowski. "Dynamic load allocation in parallel queues with synchronization." *Proceedings of the 29th Conference on Decision and Control*. 2124-2129. New York: IEEE Press, 1990.
- [57] Hackman, D. V. *Analysis of Aircraft Sortie Generation with Concurrent Maintenance and General Service Times*. M. S. thesis, Graduate School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB OH, 1997 (DTIC Number TBD).
- [58] Jenkins, R. C. *A Mean Value Analysis Heuristic for Analysis of Aircraft Sortie Generation*. M. S. thesis, Graduate School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB OH, 1994 (AD-A278578).
- [59] Kant, K. *Introduction to Computer System Performance Evaluation*. New York: McGraw Hill, Inc., 1992.
- [60] Kelley, C. T. *Iterative Methods for Linear and Nonlinear Equations*. Philadelphia: Society for Industrial and Applied Mathematics, 1995.
- [61] Kim, C. and A. K. Agrawala. "Analysis of the fork-join queue," *IEEE Transactions on Computers*, 38(2):250-255 (1989).
- [62] Kim, D. S. and R. L. Smith. "An exact aggregation/disaggregation algorithm for large scale Markov chains," *Naval Research Logistics*, 42:1115-1128 (1995).

- [63] King, R. E. *Sojourn Distributions for Particular Customers in Networks of Queues*. PhD dissertation, University of Florida, 1986 (DA8716008).
- [64] Knaup, W. "A new iterative numerical solution method for Markovian queueing networks." *Quantitative Evaluation of Computing and Communication Systems: Proceedings of the Joint Conference Performance Tools '95 and MMB '95*, edited by H. Beilner and F. Bause, 194–208. Berlin: Springer-Verlag, 1995.
- [65] Knessl, C. "On the diffusion approximation to a fork and join queueing model," *SIAM Journal of Applied Mathematics*, 51(1):160–171 (1991).
- [66] Konstantopoulos, P. and J. Walrand. "Stationarity and stability of fork-join networks," *Journal of Applied Probability*, 26(3):604–614 (1989).
- [67] Krieger, U. R. and M. Sczittnick. "Application of numerical solution methods for singular systems in field of computational probability theory." *Iterative Methods in Linear Algebra: Proceedings of the IMACS International Symposium*, edited by R. Beauwens and P. de Groen, 613–626. Amsterdam: Elsevier Science Publishers B. V. (North-Holland), 1992.
- [68] Kuehn, P. J. "Approximate analysis of general queueing networks by decomposition," *IEEE Transactions on Communications*, 27(1):113–126 (1979).
- [69] Kumar, A. and R. Shorey. "Performance analysis and scheduling of stochastic fork-join jobs in a multicomputer system," *IEEE Transactions on Parallel and Distributed Systems*, 4(10):1147–1164 (1993).
- [70] Labetoulle, J. and G. Pujolle. "Isolation method in a network of queues," *IEEE Transactions on Software Engineering*, 6(4):373–381 (1980).
- [71] Lanczos, C. "An iteration method for the solution of the eigenvalue problem of linear differential and integral operators," *Journal of Research of the National Bureau of Standards*, 45:255–282 (1950).
- [72] Lazowska, E. D., and others. *Quantitative System Performance: Computer System Analysis Using Queueing Network Models*. Englewood Cliffs NJ: Prentice Hall, Inc., 1984.
- [73] Liu, Y. and H. G. Perros. "Approximate analysis of a closed fork-join model," *European Journal of Operational Research*, 53(3):382–392 (1991).
- [74] Liu, Y. and H. G. Perros. "A decomposition procedure for the analysis of a closed fork/join queueing system," *IEEE Transactions on Computers*, 40(3):365–370 (1991).
- [75] Liu, Z. and F. Baccelli. "Generalized precedence-based queueing systems," *Mathematics of Operations Research*, 17(3):615–639 (1992).

- [76] Makowski, A. M. and R. Nelson. "An optimal scheduling policy for fork/join queues." *Proceedings of the 32nd Conference on Decision and Control*. 3611–3617. New York: IEEE Press, 1993.
- [77] Mandelbaum, M. and B. Avi-Itzhak. "Introduction to queueing with splitting and matching," *Israel Journal of Technology*, 6(5):376–382 (1968).
- [78] Marie, R. A. "An approximate analytical method for general queueing networks," *IEEE Transactions on Software Engineering*, 5(5):530–538 (1979).
- [79] Marie, R. A. "Calculating equilibrium probabilities for $\lambda(n)/C_k/1/N$ queues," *Performance Evaluation Review*, 9(2):117–125 (1980).
- [80] Marie, R. A. and J. M. Pellaumail. "Steady-state probabilities for a queue with a general service distribution and state-dependent arrivals," *IEEE Transactions on Software Engineering*, 9(1):109–113 (1983).
- [81] Marie, R. A., and others. "Extensions and computational aspects of an iterative method," *Performance Evaluation Review*, 11(4):186–194 (1982).
- [82] Meijerink, J. A. and H. A. van der Vorst. "Guidelines for the usage of incomplete decompositions in solving sets of linear equations as they occur in practical problems," *Journal of Computational Physics*, 44:134–155 (1981).
- [83] Merrill, D. Point Paper on MOG. Headquarters Air Mobility Command, Scott AFB IL, 12 December 1994.
- [84] Murata, T. "Petri nets: properties, analysis and applications," *Proceedings of the IEEE*, 77(4):541–580 (1989).
- [85] Nelson, R. and A. N. Tantawi. "Approximate analysis of fork/join synchronization in parallel queues," *IEEE Transactions on Computers*, 37(6):739–743 (1988).
- [86] Nelson, R. and A. N. Tantawi. "Approximating task response times in fork/join queues." *High Performance Computer Systems: Proceedings of the International Symposium on High Performance Computer Systems, Paris, France, 14–16 December 1987*, edited by E. Gelenbe, 157–167. Amsterdam: Elsevier Science Publishers B. V. (North-Holland), 1988.
- [87] Nelson, R. and D. Towsley. "A performance evaluation of several priority policies for parallel processing systems," *Journal of the Association for Computing Machinery*, 40(3):714–740 (1993).
- [88] Nelson, R., and others. "Performance analysis of parallel processing systems," *IEEE Transactions on Software Engineering*, 14(4):532–539 (1988).

- [89] Neuts, M. F. *Structured Stochastic Matrices of M/G/1 Type and Their Applications*. New York: Marcel Dekker, 1989.
- [90] Neuts, M. F. *Matrix-Geometric Solutions in Stochastic Models*. New York: Dover Publications, Inc., 1994.
- [91] Nguyen, V. *Heavy Traffic Analysis of Processing Networks with Parallel and Sequential Tasks*. PhD dissertation, Stanford University, 1990 (DA9108881).
- [92] Nguyen, V. "Processing networks with parallel and sequential tasks: heavy traffic analysis and Brownian limits," *Annals of Applied Probability*, 3(1):28–55 (1993).
- [93] Oppe, T. C., and others. *NSPCG User's Guide, Version 1.0: A Package for Solving Large Sparse Linear Systems by Various Iterative Methods*. Center for Numerical Analysis, University of Texas at Austin, 1988.
- [94] Paige, C. C. and M. A. Saunders. "LSQR: an algorithm for sparse linear equations and sparse least squares," *ACM Transactions on Mathematical Software*, 8(1):43–71 (1982).
- [95] Philippe, B., and others. "Numerical methods in Markov chain modeling," *Operations Research*, 40(6):1156–1179 (1992).
- [96] Rajaraman, B. and T. W. Morgan. "Approximate analysis of the average delay in parallel program execution." *Proceedings of the 26th Hawaii International Conference on System Sciences*, edited by T. N. Mudge and others, 584–593. New York: IEEE Press, 1993.
- [97] Rao, B. M. "On the departure process of the split and match queue," *Computers and Operations Research*, 17(4):349–357 (1990).
- [98] Rao, B. M. and M. J. M. Posner. "Algorithmic and approximation analyses of the split and match queue," *Communications in Statistics: Stochastic Models*, 1(2):433–456 (1985).
- [99] Rao, P. C. and R. Suri. "Approximate queueing network models for closed fabrication/assembly systems. Part I: single level systems," *Production and Operations Management*, 3(4):244–275 (1994).
- [100] Reiser, M. and S. S. Lavenberg. "Mean value analysis of closed multichain queueing networks," *Journal of the Association for Computing Machinery*, 27(2):313–320 (1980).
- [101] Rommel, C. G. "A solution for $M^X/G/1$ -PS process response time," *Operations Research Letters*, 15(5):253–263 (1994).
- [102] Ross, S. M. *Stochastic Processes*. New York: John Wiley and Sons, 1983.

- [103] Ruan, S. Y. and M. A. Jafari. "An approximation model for a manufacturing cell attended by a material handling system." *Queueing Networks with Finite Capacity: Proceedings of the International Symposium on Queueing Networks with Finite Capacity, Research Triangle Park NC, 28-29 May 1992*, edited by R. O. Onvural and I. F. Akyildiz, 225-238. Amsterdam: Elsevier Science Publishers B. V. (North-Holland), 1993.
- [104] Saad, Y. "Projection methods for the numerical solution of Markov chain models." *Numerical Solution of Markov Chains*, edited by W. J. Stewart, 455-472. Basel: Marcel Dekker, 1991.
- [105] Saad, Y. and M. H. Schultz. "GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems," *SIAM Journal of Scientific and Statistical Computing*, 7(3):856-869 (1986).
- [106] Schubert, K. A. Operations Research Analyst, Headquarters Air Mobility Command. Electronic mail message s2d67598.015@hqamc.safb.af.mil, 10 Jan 1997.
- [107] Schubert, K. A. and T. Cusick. "Base Resource and Airfield Capability Evaluation (BRACE) Simulation Model: System Documentation." Unpublished paper. Headquarters Air Mobility Command, Scott AFB IL, October 1996.
- [108] Schweitzer, P. J. "A survey of aggregation-disaggregation in large Markov chains." *Numerical Solution of Markov Chains*, edited by W. J. Stewart, 63-88. Basel: Marcel Dekker, 1991.
- [109] Setia, S. K., and others. "Analysis of processor allocation in multiprogrammed, distributed-memory parallel processing systems," *IEEE Transactions on Parallel and Distributed Systems*, 5(4):401-420 (1994).
- [110] Shioyama, T. and K. Tanaka. "A new aggregation-disaggregation algorithm," *European Journal of Operational Research*, 83(3):655-669 (1995).
- [111] Sonneveld, P. "CGS, a fast Lanczos-type solver for nonsymmetric linear systems," *SIAM Journal of Scientific and Statistical Computing*, 10(1):36-52 (1989).
- [112] Steppe, J. M. Operations Research Analyst, Headquarters Air Mobility Command. Written communication, 5 Aug 1995.
- [113] Steppe, J. M. Operations Research Analyst, Headquarters Air Mobility Command. Personal interview, 26 May 1995.
- [114] Steppe, J. M. "BRACE: Base Resource and [Airfield] Capability Evaluation." Presentation to the 63rd Military Operations Research Society Symposium, Annapolis MD, 6-8 June 1995. Headquarters Air Mobility Command, Scott AFB IL, 6 June 1995.

- [115] Steppe, J. M., and others. "Stochastic Airfield Capability Modeling." Unpublished paper. Headquarters Air Mobility Command, Scott AFB IL, February 1995.
- [116] Stewart, W. J. *An Introduction to the Numerical Solution of Markov Chains*. Princeton NJ: Princeton University Press, 1994.
- [117] Stewart, W. J. and R. A. Marie. "A numerical solution for the $\lambda(n)/C_k/r/N$ queue," *European Journal of Operational Research*, 5(1):56-68 (1980).
- [118] Stewart, W. J. and W. Wu. "Numerical experiments with iteration and aggregation for Markov chains," *ORSA Journal on Computing*, 4(3):336-350 (1992).
- [119] Stoyan, D. *Comparison Methods for Queues and Other Stochastic Models*. Chichester UK: John Wiley and Sons, 1983. Translated from the German and revised by D. J. Daley.
- [120] Thomasian, A. and A. N. Tantawi. "Approximate solutions for M/G/1 fork/join synchronization." *Proceedings of the 1994 Winter Simulation Conference*, edited by J. D. Tew and others, 361-368. New York: IEEE Press, 1994.
- [121] Tijms, H. C. *Stochastic Models: An Algorithmic Approach*. Chichester UK: John Wiley and Sons, 1994.
- [122] Touzene, A. "A new iterative method for solving large-scale Markov chains." *Quantitative Evaluation of Computing and Communication Systems: Proceedings of the Joint Conference Performance Tools '95 and MMB '95*, edited by H. Beilner and F. Bause, 180-193. Berlin: Springer-Verlag, 1995.
- [123] Towsley, D., and others. "The performance of processor sharing for scheduling fork-join jobs in multiprocessors." *High Performance Computer Systems: Proceedings of the International Symposium on High Performance Computer Systems, Paris, France, 14-16 December 1987*, edited by E. Gelenbe, 145-156. Amsterdam: Elsevier Science Publishers B. V. (North-Holland), 1988.
- [124] Towsley, D., and others. "Analysis of fork-join program response times on multiprocessors," *IEEE Transactions on Parallel and Distributed Systems*, 1(3):286-303 (1990).
- [125] van der Vorst, H. A. "The convergence behavior of preconditioned CG and CG-S in the presence of rounding errors." *Preconditioned Conjugate Gradient Methods: Proceedings of a Conference Held in Nijmegen, The Netherlands, June 19-21, 1989*, edited by O. Axelsson and L. Yu. Kolotilina, 126-136. Berlin: Springer-Verlag, 1990.

- [126] van der Vorst, H. A. "Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems," *SIAM Journal of Scientific and Statistical Computing*, 13(2):631–644 (1992).
- [127] Varma, S. "Performance evaluation of the time-stamp ordering algorithm in a distributed database," *IEEE Transactions on Parallel and Distributed Systems*, 4(6):668–676 (1993).
- [128] Varma, S. and A. M. Makowski. "Interpolation approximations for symmetric fork-join queues," *Performance Evaluation*, 20(1–3):245–265 (1994).
- [129] Waisanen, A. and M. Grabau. "Airfield resource modeling (ARM): a mobility analysis tool." Unpublished paper. Headquarters Air Mobility Command, Scott AFB IL, 1992.
- [130] Walker, H. F. "Implementation of the GMRES method using Householder transformations," *SIAM Journal of Scientific and Statistical Computing*, 9(1):152–163 (1988).
- [131] Whitt, W. "Approximating a point process by a renewal process, I: two basic methods," *Operations Research*, 30(1):125–147 (1981).
- [132] Zhang, Z. "Analytical results for waiting time and system size distributions in two parallel queueing systems," *SIAM Journal on Applied Mathematics*, 50(4):1176–1193 (1990).

Vita

Craig Joslyn Willits [REDACTED]

He attended public schools in Rumson NJ, graduating from Rumson-Fair Haven Regional High School in 1979. In 1983, he received a Bachelor of Arts degree with a major in mathematics from Rutgers, The State University of New Jersey.

An Air Force Reserve Officers Training Corps Distinguished Graduate, Major Willits received a regular Air Force commission after graduation from Rutgers. From October 1983 to August 1992, he performed a variety of analysis duties, as well as serving a tour as an aircraft maintenance officer. During this period, he was stationed at Los Angeles AFB CA, Chanute AFB IL, March AFB CA, and Barksdale AFB LA. In August 1992, Major Willits entered the Air Force Institute of Technology. He completed the Graduate Operations Research program in March 1994 as a Distinguished Graduate, and was awarded a Master of Science degree in operations research by the Institute. After receiving his master's degree, Major Willits remained at the Institute as a doctoral student in operations research.

Major Willits is a member of the Omega Rho and Tau Beta Pi honor societies and the Institute for Operations Research and the Management Sciences. His master's thesis, "Point and Interval Estimation of Series System Reliability Using Small Data Sets," won the Military Operations Research Society's 1994 Graduate Research Award. He has published an archival journal article and a refereed proceedings article based on his master's thesis research.

Major Willits and his wife, the former Sara Elizabeth Brown of Hackettstown, New Jersey, have three sons: Steven, David and Joseph.

[REDACTED]

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE March 1997		3. REPORT TYPE AND DATES COVERED Doctoral Dissertation
4. TITLE AND SUBTITLE NESTED FORK-JOIN QUEUING NETWORKS AND THEIR APPLICATION TO MOBILITY AIRFIELD OPERATIONS ANALYSIS			5. FUNDING NUMBERS	
6. AUTHOR(S) Craig Joslyn Willits, Major, USAF				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology Wright-Patterson AFB OH 45433-7765			8. PERFORMING ORGANIZATION REPORT NUMBER AFIT/DS/ENS/97-01	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) AMCSAF/XPYA 204 SCOTT DR UNIT 3L3 SCOTT AFB IL 62225-5307			10. SPONSORING/MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution unlimited			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) A single-chain nested fork-join queuing network (FJQN) model of mobility airfield ground processing is proposed. In order to analyze the queuing network model, advances on two fronts are made. First, a general technique for decomposing nested FJQNs with probabilistic forks is proposed, which consists of incorporating feedback loops into the embedded Markov chain of the synchronization station, then using Marie's Method to decompose the network. Numerical studies show this strategy to be effective, with less than two percent relative error in the approximate performance measures in most realistic cases. The second contribution is the identification of a quick, efficient method for solving for the stationary probabilities of the $\lambda(n)/C_k/r/N$ queue. Unpreconditioned Conjugate Gradient Squared is shown to be the method of choice in the context of decomposition using Marie's Method, thus broadening the class of networks where the method is of practical use. The mobility airfield model is analyzed using the strategies described above, and accurate approximations of airfield performance measures are obtained in a fraction of the time needed for a simulation study. The proposed airfield modeling approach is especially effective for quick-look studies and sensitivity analysis.				
14. SUBJECT TERMS Queuing Networks, Decomposition, Markov Chains, Steady State Analysis, Mobility Modeling, Large Linear Systems			15. NUMBER OF PAGES 190	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL	

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to ***stay within the lines*** to meet ***optical scanning requirements***.

Block 1. Agency Use Only (Leave blank).

Block 2. Report Date. Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C - Contract	PR - Project
G - Grant	TA - Task
PE - Program Element	WU - Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Name(s) and Address(es). Self-explanatory.

Block 10. Sponsoring/Monitoring Agency Report Number. (If known)

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with...; Trans. of...; To be published in.... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

DOD - See DoDD 5230.24, "Distribution Statements on Technical Documents."

DOE - See authorities.

NASA - See Handbook NHB 2200.2.

NTIS - Leave blank.

Block 12b. Distribution Code.

DOD - Leave blank.

DOE - Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.

NASA - Leave blank.

NTIS - Leave blank.

Block 13. Abstract. Include a brief (*Maximum 200 words*) factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code (*NTIS only*).

Blocks 17. - 19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.